USER MANUAL

VECTIS-MAX Version 1.1

December 2009

Contents

List of Tables

List of Figures

1 USING THIS MANUAL

The latest versions of this manual (including pdf version) and the release notes are available at Vectis-Max Documents.

1.1 Contents of This Manual

The manual provides all necessary information in order to use VECTIS-MAX effectively. A brief description of the contents of each chapter is given as follows:

- □ Introduction outlines the main features and capabilities of VECTIS-MAX . A brief guide is also provided which explains how to start, run and analyse the VECTIS-MAX simulation.
- □ Geometry related chapter is the description of *phase1* . It deals with the geometry acquisition and its preparation for the meshing task. The definition of boundary regions (open flow boundaries and walls) is also described.
- □ Meshing, i.e. generation of the numerical mesh, for single– and multi–domain simulations is described in this chapter. Here, the user can learn how to create the VECTIS-MAX mesh using the *vsolve* module or to import non–VECTIS-MAX meshes. The mesh structure (connectivity data) used by the solver as well as mesh quality checks are presented.
- □ Reading and Manipulating meshes chapter describes usual pre–processing tasks needed for the solver run. This includes mesh partitioning in case of parallel runs, enabling partitioning for the solver restarts and creating conformal meshes at fluid–solid interfaces for multi–domain (CHT) simulations. All these tasks are done with the *vpre* module.
- □ Solver Fundamentals contains mathematical background: continuum conservation equations, their closure problem and averaging leading to the Reynolds–averaged equations.
- □ Modelling Spatial Domains chapter provides information about multi–domain structure and how to create this structure.
- □ Modelling Continua and Their Properties describes thermo–physical properties of fluid and solid continua, including single– and multi–component phases. The calculation options for various properties, the corresponding equations and setting of the properties are provided.

- Modelling Turbulence chapter introduces the turbulence problem, gives an overview of the modelling approaches and describes currently implemented models and how to select them.
- Modelling Single–Phase Flows describes the available single–phase physical models and their selection/set up. Separate sections present modelling of fluid flows, heat transfer in fluids and solid as well as conjugate heat transfer and mass transfer.
- Modelling Porous Media chapter deals with three–dimensional regions (fluid sub–domains) occupied by continuum which comprises both fluid material and fine–scale solid structure. It contains theoretical background relevant to the volume–averaged porous media equations and description of currently available porous types and their definition by the user.
- Modelling Multiphase Flows chapter introduces the physical aspects and modelling approaches for multi–phase flows. It describes in details multi–fluid Euler–Euler and single–fluid Mixture and VOF models and how to use the currently available homogeneous mixture model, boiling and cavitation models.
- Boundary Condition and Interface Types available in VECTIS-MAX are described in this chapter. Their physical aspects are explained as well as their definition.
- Numerical Solution of the governing equations is described here. The user inputs, required at the different stages of an iterative solution procedure are explained.
- Modelling Radiation describes the surface–to–surface radiation module, its individual programs (*radvfm* , *radsolv*) and its interface to VECTIS-MAX . In particular, attention is paid to the mesh file and radiation setup.
- Modelling Fans chapter presents pragmatic and computationally inexpensive modelling practice for fluid flows through fans. Two models, labelled as 1D and sub–domain, are available, and both require the specification of a pair of inlet/outlet boundaries or sub–domain interfaces.
- Using Solver chapter is intended to provide the user with information not contained in the previous chapters: how to monitor the solution, use the restart control, interact with the solver, do WAVE –VECTIS-MAX co–simulation and understand the VECTIS-MAX output and alpha–numeric reports.
- User Programming tells the users how to write and compile their own Fortran 95/2003 source codes. Various tasks can be performed such as the definition of boundary and initial conditions, material properties or output results.
- Tutorials are the starting point for those who want to try VECTIS-MAX immediately, without reading other chapters of this manual. This chapter contains the basic tutorial followed by two application examples (steady–state port and coolant flows), and by mesh import tutorial.
- The manual also contains a bibliography used in various chapters.

1.2 Other Manuals

Currently there are no other manuals related to VECTIS-MAX .

1.3 Acknowledgements

VECTIS-MAX uses the following 3rd party software:

- MPI - Message Passing Interface `http://www.mpi-forum.org`
- METIS - Graph Partitioning from Karypis Lab, University of Minnesota `http://glaros.dtc.umn.edu/gkhome/views/metis`
- TETGEN - Tetrahedral Mesh Generator from Weierstrass Institute `http://www.wias-berlin.de`
- CGNS - CFD General Notation System Steering Committee `http://www.grc.nasa.gov/WWW/cgns/charter/index.html`
- HDF5/SZIP - The HDF Group, University of Illinois `http://www.hdfgroup.org/HDF5/`

2 INTRODUCTION

VECTIS-MAX is a next generation computational fluid dynamics (CFD) software product developed by Ricardo. It is a general purpose tool for solving advanced 3D industrial fluid flow and heat transfer problems, with particular attention given to the requirements of automotive applications.

The product contains a fully automatic Cut–Cartesian mesh generator, a multi–domain solver capable of running on arbitrary unstructured meshes, and an advanced Graphical User Interface (GUI) to prepare geometry and display simulation results. Typically, the multi–domain features are used to simulate Conjugate Heat Transfer (CHT) in automotive applications such as cooling of cylinder heads and engine blocks.

2.1 Main Features and Capabilities

An overview of the VECTIS-MAX capabilities is given below by considering the three key software modules.

2.1.1 Meshing – *vmesh*

- □ *Rapid and automatic vmesh generator.* The *vmesh* program is a fully automatic hexahedral mesh generator, producing Cut–Cartesian meshes using a hybrid scheme of Exact Fit and Marching Cubes. The ability to mesh easily complex CAD geometries, without resorting to the boundary surface grid generation, is the main advantage of *vmesh* . The program has a number of improvements over the previous mesh generator, taking advantage of a clean sheet design and the unstructured grid capability of the new solver. For example separated volumes that are found in one box are now stored as separate cells. Boundary faces are now tied exactly to each other. Automatic scaling of the input geometry now prevents problems with very large and very small geometries. Faster meshing times are now achieved because of advanced techniques for storing and sorting data, with the additional benefit of reduced memory consumption.
- □ *Conformal meshing for multi-domain modelling.* The mesher is able to produce separate meshes for each domain that are conformal at the domain interfaces. These meshes are then coalesced into a single multi-domain grid file using the *vpre* utility.

- □ *Cell splitting.* This feature deals with highly non–convex cells. It is automatically activated when running *vmesh* . Each non-convex cell is split into two cells with better (less concave) shape.
- □ *Geometry (surface) import.* Import of ASCII and binary VECTIS-MAX *.tri* triangle files, VDA files and Stereo-lithography *.stl* files. The import of geometry is done by using *phase1* module.

2.1.2 Solver – *vsolve*

The *vsolve* program is a state of the art solver using advanced CFD algorithms.

Design Concepts

- □ *Modular code.* There are two major parts: *CFD kernel* and *Application modules*
- □ *Data structure.* The data structure and software design of VECTIS-MAX kernel supports multi–domain and multi–physics modelling of a general multi–phase and/or multi–component continuum.
- □ *Unstructured grid solver.* The ability to perform CFD simulations on an unstructured grid, either generated with the native mesher *vmesh* or imported from a 3rd party format (CGNS, Ideas universal .unv, etc.) All mesh types are accommodated by general polyhedral control volume formulation.

Continua Properties

Thermo–physical and thermodynamics property specification is designed to be as general as possible with properties specified for each species and phase separately using a range of options. These options include fixed values, ideal gas, temperature dependent polynomials or exponentials. Properties can also be specified using user functions or from WAVE property files.

Numerical Method

- □ *Discretization.* Advanced finite–volume based discretization applicable to arbitrary polyhedral cells.
- □ *Linear equations solvers.* Both bi-conjugate gradient and symmetric conjugate gradient linear solvers are available with a number of pre-conditioning options including incomplete Lower Upper, Modified Stone's SIP, and Jacobi.
- □ *Pressure–velocity–density coupling.* Segregated implicit SIMPLE–like approach for all–speed incompressible and compressible flows.
- □ *Poor quality mesh (cells).* A novel cell–interpolative scheme deals with low quality cells that can be produced by *vmesh* and any other mesher. In case of *vmesh* the low quality cells are typically produced when the insufficient mesh refinement can not resolve fine geometry features.

- □ *Conjugate Heat Transfer (CHT).* Novel and efficient techniques are introduced for the implicit solution of the energy equation. The CHT simulations require conformal grids at fluid/solid or solid/solid interfaces.

Parallelisation

Scalable parallelisation of VECTIS-MAX is based on the domain decomposition strategy and MPI-based message passing. Domain decomposition is performed using METIS to decompose the mesh into partitions taking into account load balancing for each material for maximum efficiency. The number of partitions used for the solution can be changed during a calculation by utilising the re-partitioning of restart files feature of the *vpre* module.

Boundary and Interface Condition Types

- □ *Wall and flow boundaries.* In addition to walls and symmetry planes, various flow boundary condition types can be specified as fixed static, average or total pressure at inlet or outlet, inlet velocity and mass flow rate. Inlet stagnation conditions are also available along with an outlet boundary condition.
- □ *Wall thermal conditions.* For the solution of the energy equation various thermal conditions can be specified: the wall temperature or wall heat flux, external convective heat transfer, external radiation heat transfer and combination of external convective and radiation heat transfer. In conjunction with any of the above thermal conditions the wall thickness can be taken into account using thin–wall model.
- □ *R–Therm* . The wall thermal conditions can be imported via *R–Therm* model, i.e. using "*R–Therm* " files.
- □ *Interface conditions.* For fluid flow equations, with exception of the energy, an interface condition type is the same as the wall boundary type. For the energy equation all interfaces are treated in an implicit manner, i.e. no specific boundary types are required.

Basic Physical Models

VECTIS-MAX provides a range of physical models for the single or multi–component phase in terms of:

- □ *Equation of state and compressibility.* Incompressible substance (liquids, gases and solids) and ideal gas model are available. All speed–flow flows (incompressible or weakly compressible, subsonic and supersonic) can be simulated.
- □ *Dimensionality.* Two– and three–dimensional simulations can be performed.
- □ *Single or multi–component phase.* For either single-phase or multi-phase fluid flow the fluid phase can be selected as a single-component or multi-component. The multi–component phase is a mixture of species.

- ☐ *Flow regimes.* Both steady and unsteady simulations of inviscid, laminar or turbulent flows can be performed. The potential flow model is also available as a part of the flow initialisation.
- ☐ *Body forces.* Modelling of the gravity force effects is provided for all flow regimes.
- ☐ *Heat transfer.* All convection modes - forced, natural and mixed - can be simulated by solving momentum and energy equations in fluids. In solids, the heat conduction is modelled. CHT modelling, describing coupled heat transfer through adjacent fluid and solid material domains, is also available.
- ☐ *Mass transfer.* In case of multi–component phase, which is defined as a mixture of species, modelling of mass transfer for *non–reacting* flows is supported.
- ☐ *Passive scalars.* Similar to standard species, transport of passive scalars can be modelled.

Turbulence Modelling

The RANS (Reynolds Averaged Navier–Stokes) approach is currently employed and closure of the RANS equations is provided via linear two-equation $k-\varepsilon$ models: standard, standard with realisable time–scale option and RNG. In the near–wall regions viscous and wall blocking effects are accounted for by using either standard or scalable or enhanced *wall functions*.

Multi–Phase Physical Models

The current version of VECTIS-MAX solver includes two models:

- ☐ *Homogeneous mixture model.* This model neglects the relative motion between phases. In addition, other relevant fields such as temperature and turbulence quantities are shared by all phases.
- ☐ *Nucleate boiling model.* Two nucleate boiling models, VECTIS3 and *RPI*, designed to predict boiling possibly taking place in engine cooling passages, are provided. Both models belong to the homogeneous mixture family.

Surface–to–Surface Radiation

The surface–to–surface radiation module, applicable to fluid domains and containing the *radprep* and *radvfm* sub–modules, is directly coupled to the VECTIS-MAX solution of the energy equation.

User Programming

A powerful set of User Programmable Routines (UPR, Fortran 95/2003) have been implemented that allow direct access into the solver kernel data structure's. These functions allow the user to perform many tasks such as setting up boundary conditions, initial conditions, material properties or adding source terms.

Other User Tools

The user can monitor and/or interact with the solution process in a number of ways:

- □ *Monitors.* Flow variable values can be monitored at any number of user specified locations.
- □ *Alpha–numerical reports.* These reports, containing typically variable average values, are available for fluid/solid domains, boundary and interface regions and for user defined arbitrary surface.
- □ *"Live update".* The "Live Update" from *R–Desk* can be used to monitor the solution convergence. The equations residuals or averaged variables over boundary regions or over domains are displayed against the solver iterations/time steps.
- □ *Run–time solution control.* Run time control of the solver allows controls such as changing the solution end time, changing output file writing frequencies or saving immediately the post–processing file.

Interface to WAVE

It is possible to run coupled co-simulations with the Ricardo WAVE 1D gas flow product, where the CFD inflow and/or outflow boundary conditions are supplied from a WAVE calculation. Non-flow boundaries can either be specified as symmetry boundaries or walls, with fixed heat flux or temperature.

2.1.3 Pre– and post–processing: *R–Desk*

All pre and post–processing utilities for CFD analysis are integrated into the new Ricardo GUI *R–Desk* . This product introduces many new capabilities including multiple viewports and combined display of CFD results and structural analysis results together. Further details of the features are described in the *R–Desk* Help

2.2 Using VECTIS-MAX – Brief Guide

The following section provides a brief guide to the setting-up process (from start to finish) for a typical CFD simulation within VECTIS. Figure (2.1) illustrates this process.

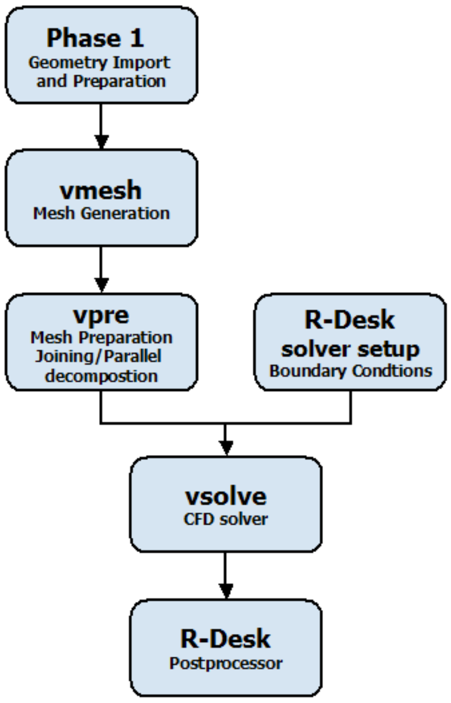


Figure 2.1: Vectis work-flow - from initial model geometry to viewing CFD results.

The work-flow using VECTIS consists of several stages, beginning with:

Geometry Import and Preparation

The is performed using the Phase 1 Preprocessor. This has a number of main functions:

1. To read in geometry from an external source and convert it to a form acceptable for the VECTIS mesh generator (vmesh).
2. To identify regions of the geometry as different boundaries for the CFD calculation.
3. To set up the "global mesh" and other control parameters for vmesh.
4. To start up the mesh generator, and view the resulting mesh.

Mesh Generation

Vmesh is the VECTIS mesh generator. It is fully automatic, and produces a locally refined Cartesian mesh which is suitable for fluid flow analysis using the VECTIS solver (vsolve). The input to the mesh generator is a completely closed, fully connected triangulated surface, which contains boundary information, as described in the Phase1 section. Phase 1 is used to supply this surface and a further file containing the meshing control parameters. Vmesh writes the computational grid (.GRD) file which contains information about the mesh geometry, and its associated boundary faces.

Mesh Preparation

The vpre mesh preparation stages allows further manipulations of the mesh generated by the vmesh program.

1. Mesh sub-division into separate partitions for parallel calculations.
2. Conformal joining of grid files.
3. Importing (and converting) external mesh files.

Solver Setup

A controlling .inp input file needs to be defined for the CFD solver (vsolve). This is created in R-Desk using the 'solver setup' panels. The main parameters specified by the input file include:

- ☐ Selected time base to be steady or unsteady.
- ☐ Time step and convergence criteria.
- ☐ Specify monitoring and reference locations.
- ☐ Frequency of restart and post-processing file dumps.
- ☐ Restart options such as whether to start from an existing restart file or not.
- ☐ Fluid/solid property data.
- ☐ Equations to be solved.
- ☐ Numerical schemes used to solve the equations.
- ☐ Boundary conditions.
  - Walls.
  - Inlet/Outlets.
- ☐ Initial conditions.

CFD Solver

Vsolve is the solving stage of VECTIS - this is the phase that actually performs the computational fluid dynamics calculation. The CFD solver uses the generated computational grid file and the input file.

Post-processing

The results produced by the CFD computation and written to the "post" file can be viewed in the R-Desk post-processor. This allows the simulation data to be visualised, manipulated and extracted for further use.

3 GEOMETRY

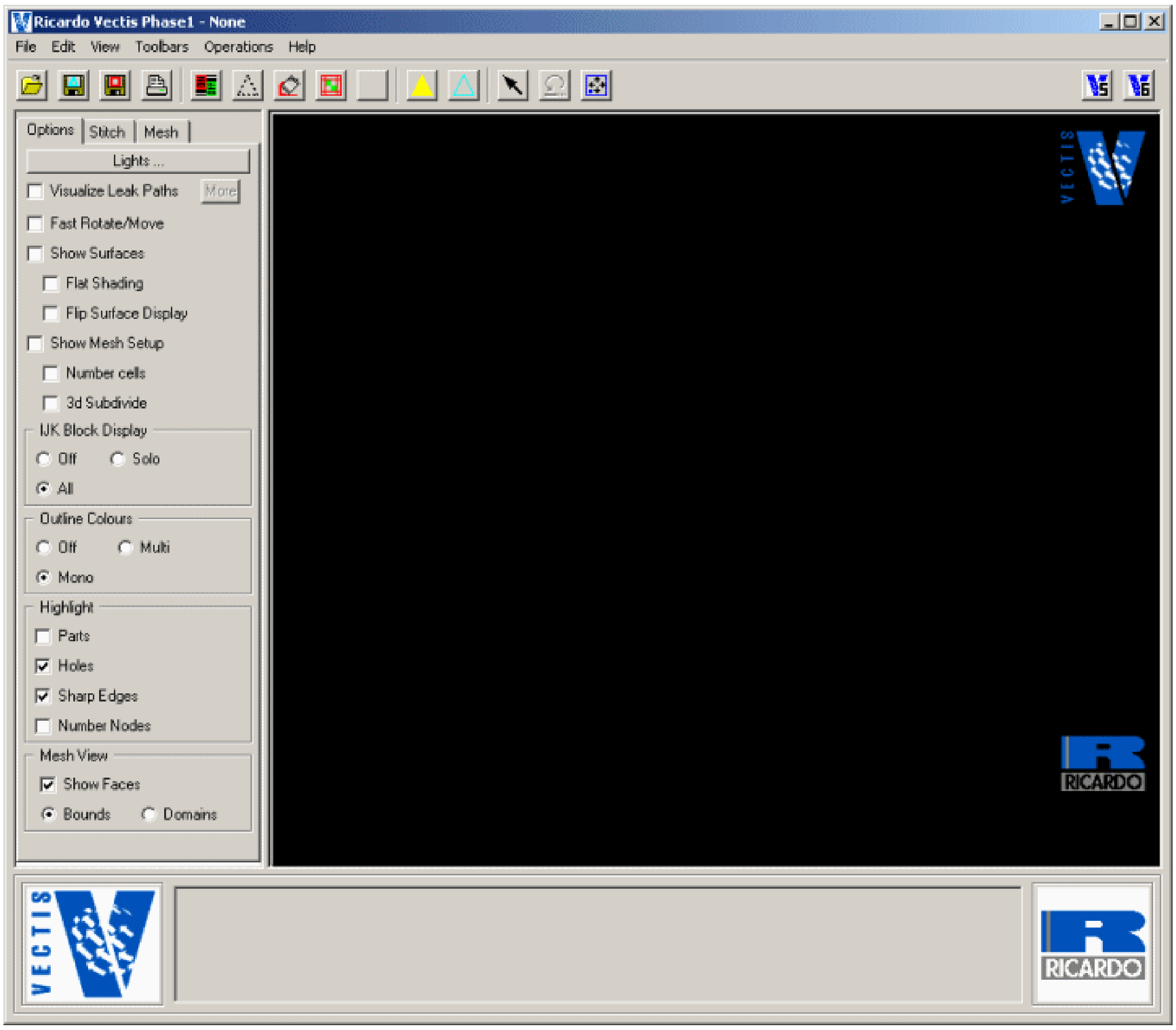
3.1 Introduction

Phase 1 has a number of main functions:

1. To read in geometry from an external source and convert it to a form acceptable to the VECTIS mesh generator.
2. To identify regions of the geometry as different boundaries for the CFD calculation.
3. To set up multiple geometry positions for use in moving mesh calculations.
4. To set up the "global mesh" and other control parameters for the mesher.
5. To start up the mesh generator, and view the resulting mesh.

3.2 User Interface

The Phase 1 GUI consists of the main menus, the button bar, the OpenGL canvas, the information area and the tool panels.


Ricardo Vectis Phase1 - None
File Edit View Toolbars Operations Help
Options Stitch Mesh
Lights ...
More
VECTIS
RICARDO

☐ Visualize Leak Paths
☐ Fast Rotate/Move
☐ Show Surfaces
☐ Flat Shading
☐ Flip Surface Display
☐ Show Mesh Setup
☐ Number cells
☐ 3d Subdivide

IJK Block Display
○ Off ○ Solo
● All

Outline Colours
○ Off ○ Multi
● Mono

Highlight
☐ Parts
☑ Holes
☑ Sharp Edges
☐ Number Nodes

Mesh View
☑ Show Faces
● Bounds ○ Domains

3.2.1 The Button Bar commands

Open File Button

Model files and mesh files can be opened using the file open button.

Save Geometry Button

Geometry can be saved using the 'Save Geometry' button

Save Mesh Set-up Button

Mesh set-up files can be saved via the 'Save Mesh Set-up' button.

Print button

The print button pops up the print panel.

The Toolset Buttons

The 'View Options', 'Stitch Tools', 'Boundary Painting', 'Mesh Set-up Tools' and 'Mesh View Tools' buttons can be used to switch between the modes and pop up the respective panels.

Toggle Surface Display Button

The toggle surface display button switches the triangle surface display on/off.

Toggle Line Display Button

The toggle line display button switches the triangle edge display on/off.

Cancel Command Button

The cancel command button cancels any current command selected from the tool panels and restores the cursor.

Undo Button

The undo button undoes the action of the previous command.

View Reset Button

The view reset button resets the view orientation and re-centres the model.

3.3 Graphics Interaction

Manipulation of the model image on the graphics canvas can be performed via the mouse or the keyboard.

Model translation

The middle mouse button is used to alter the view of the model. On its own, the middle mouse button is used for translation; holding the button down and moving the mouse makes the model move with the mouse.

Model rotation

If the SHIFT key is held down at the same time as the middle mouse button, it performs rotations. With the mouse pointer near the centre of the window, moving horizontally rotates about the y axis and moving vertically rotates about the x axis. Near the edge of the window, tangential movements rotate about z, whilst radial movements rotate about x and/or y as appropriate. The centre and edge behaviours blend smoothly into one another.

Model zooming

In conjunction with the CTRL key, the middle mouse button performs zooming; moving up and to the right zooms in and moving down and to the left zooms out.

The right mouse button is used to perform zooming and translation. If the button is depressed, and released in the same place, that point is moved to the centre of the window. If, however, the mouse button is held down while the mouse is moved, the user can drag out a rectangle with one corner at the point where the mouse was first depressed. The rectangle has the same aspect ratio as the graphics window, and defines the area that will be visible when the mouse button is released. Thus the user can zoom in very rapidly on a chosen portion of the screen.

Keyboard shortcuts

The manipulations described above can also be accomplished from the keyboard:

F1 and F2	x-rotation
F3 and F4	y-rotation
F5 and F6	z-rotation
F7 and F8	x-translation
F9 and F10	y-translation
F11 and F12	Zoom

The near and far clipping planes are moved with the following keyboard keys:

U key	Near plane outwards.
I key	Near plane inwards.
O key	Far plane inwards.
P key	Far plane outwards.

Holding down the SHIFT key while moving the clipping planes moves both planes at once, thus allowing the user to traverse a "slice" through a model.

The default rotation increment produced by pressing a function key is 45 degrees. The sensitivity can be adjusted with the numeric keypad keys as follows:

Key 0	45 degrees.
Key 1	1 degree.
Key 2	5 degrees.
Key 3	15 degrees.

The sizes of the translations and zooming scale accordingly.

The Esc key can be used to cancel any current commands requested from any of the tool panels. This may also be done using the cancel command button located on the button bar.

Holding down the CONTROL key and pressing "r" resets the model view. This can also be achieved using the view reset button located on the button bar. In the reset view operation the active part of the model is re-centred and the axis orientation is reset to an orthogonal state.

3.4 Model Import

3.4.1 Model Import

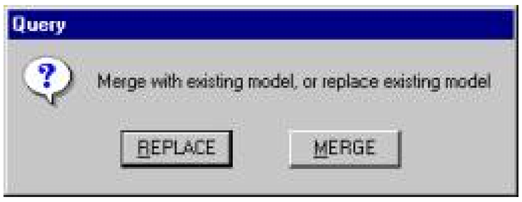
Phase 1 currently understands the following file formats:

- ASCII VECTIS Triangle file (before version 3.2.2)
- binary VECTIS Triangle file (version 3.2.2 and later)
- VDA File
- STL File (both binary and ASCII)
- Phase 4 output mesh file (viewing only)

- VECTIS mesh input file (will be associated with the current model)

All of these file formats may be imported by selecting the Open option from the File menu. Additionally, on those platforms that support a command line, a filename specified upon the command line from which Phase 1 is launched will be opened.

If the file to be loaded is a VECTIS Triangle file (of either format), the choice of whether to replace any existing model with the new model or to merge the new with the old will be given.

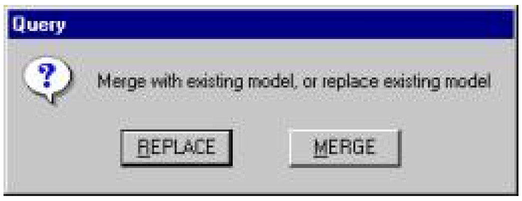

Query

Merge with existing model, or replace existing model

Merging or replacing

3.4.2 ASCII Triangle Files

Once Phase 1 has determined that an ASCII Triangle file is to be imported, the program will prompt for the location of the associated "sets" and "types" files.


Query

Merge with existing model, or replace existing model

ASCII Triangle file import

Either or both of these text fields may be left blank if required.

3.4.3 VDA File Triangulation

Phase 1 reads VDA-FS files; these can be read and written by most CAD systems. The information of interest to VECTIS in these files is the polynomial surfaces, which may be trimmed or untrimmed.

The geometrical elements handled by Phase 1 are:

SURF Polynomial surfaces.

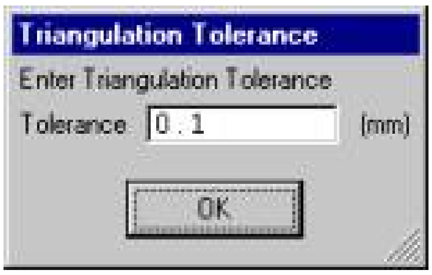
CONS Trim lines defined on a SURF element.

FACE Trimmed surfaces defined by a SURF and one or more CONS elements.

The user is referred to the VDA standard documentation for details of how these elements are defined.

VECTIS converts these surfaces to a triangulated approximation, with a user-specified accuracy which ensures that all triangle vertices lie on the original surface, and all triangle side midpoints lie within the given tolerance of the surface.

The user is prompted for the triangulation tolerance. The default is 0.1 mm.


Triangulation Tolerance

Enter Triangulation Tolerance

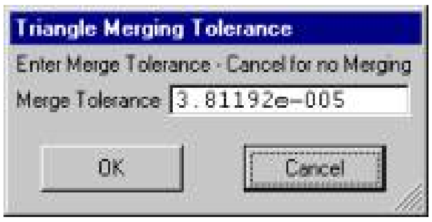
Tolerance 0.1 (mm)

VDA Triangulation Tolerance

Reading and triangulating the file may take several minutes for a large model.

3.4.4 Merging

Once a file has been read into Phase 1, the nodes in the model are merged to eliminate duplicate, or very close nodes. The program prompts the user for the tolerance for this merging. The default tolerance is such that the merging will eliminate no triangles (i.e. it is less than the shortest triangle side in the model).


Triangle Merging Tolerance

Enter Merge Tolerance - Cancel for no Merging

Merge Tolerance 3.81192e-005

Node Merging

The triangle merging process also links the imported triangles into a surface. Therefore, it is recommended that the OK button is chosen from this dialog, even if no triangles will be eliminated.

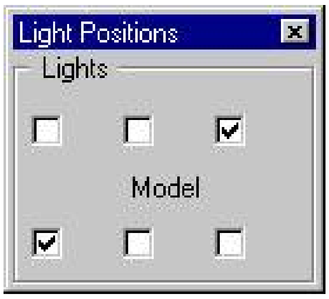
3.5 General View Options

Once a model has been loaded into Phase 1, the way in which it is viewed may be modified. The viewing options appear to the left of the drawing canvas when the application first starts up.

Selecting Options from the View menu will bring this panel back up. Alternatively the user can select the view tab or the view options button located on the button bar.

Lights

Pressing the Lights button will bring up a new panel that allows lights to be turned on and off. The lights are placed in default positions around the model.



The position of the tick-box with respect to the word "Model" corresponds to the location of the light with respect to the model on the screen.

Fast Rotate/Move

Selecting 'Fast Rotate/Move' switches on the fast rendering of the model when mouse rotate and move operations are being performed. In this mode a fraction of the model triangles are drawn allowing the screen to refresh at a much faster rate.

Show Surfaces

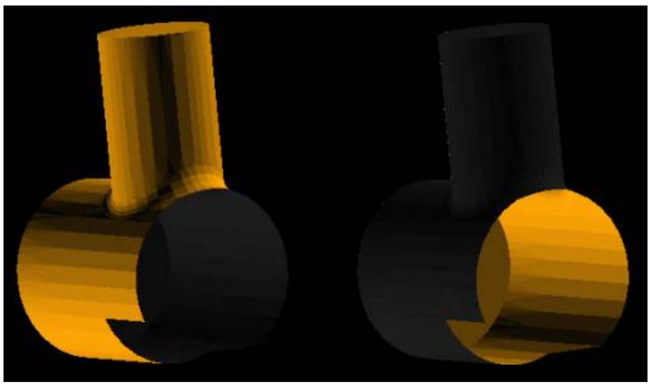
Selecting the 'Show Surfaces' option will cause the triangulated surface of the model to be shown. By default, the surface is shaded in the colour assigned to the boundary that the triangles belong to and as if lit by the lights defined above.

Flat Shading

If the 'Flat Shading' option is chosen, the lighting is ignored and the triangles are simply filled with the appropriate colour.

Flip Surface Display

The ’Flip Surface Display’ option causes the triangles to appear as if they were lit from the other side. This may help to show the detail of cut interior surfaces or to light the model properly if it has been defined with the surface normal facing in the opposite direction to that which Phase 1 expects.



Outline Colours

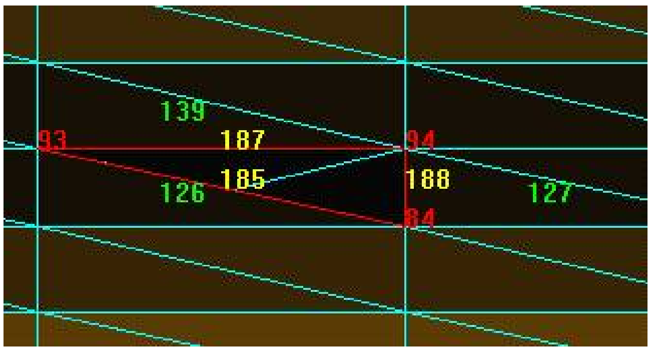
The ’Outline Colours’ option draws an outline around each of the triangles defined in the model. These outlines can be switched off or displayed as mono-coloured lines (blue) or multi-coloured lines (boundary colours)

Highlight Parts

The ’Highlight Parts’ toggle can be used to identify parts defined in the part tree shown in the Part Boundary Panel. When part highlighting is switched on selecting a part in the part tree with the mouse cursor will cause the part to be highlighted in red.

Highlight Holes and Sharp Edges

The ’Highlight Holes’ and ’Highlight Sharp Edges’ toggles highlight potential defects in the model. Holes are indicated by outlining the edges of the triangles that border them in red. The finished model must be free from holes. Sharp triangles are defined as adjacent triangle pairs that have an angle of more than 170 degrees between their normals. The edge that these triangles share is outlined in green.



Number Nodes

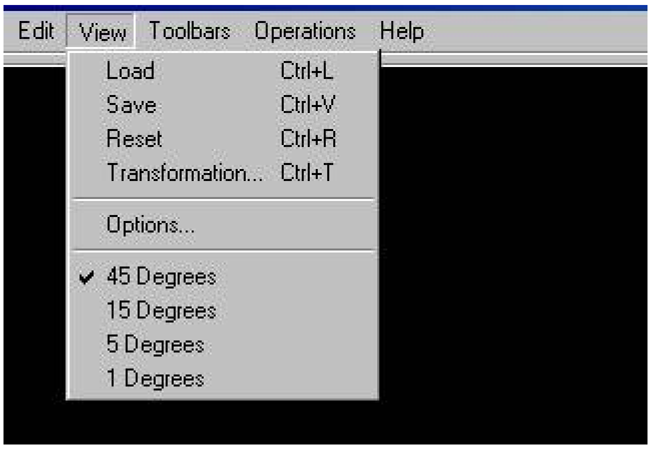
’Number Nodes’ actually annotates nodes, triangles, and edges that border a hole in the surface with a unique reference number, as shown below. The numbers of the nodes are shown in red, those of the triangles in green, and the numbers associated with the lines are in yellow.

Config File

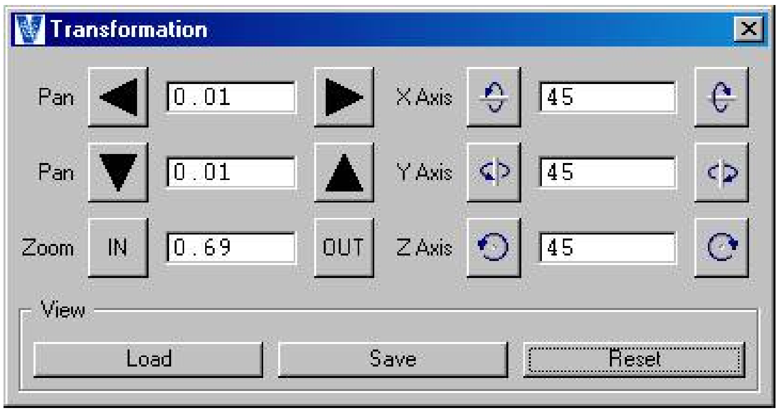
The view options can also be set in the Vectis.cfg file which is read at program start.

3.5.1 View Menu

The load and save operations available from the View Menu allow a view to be saved to disk or recovered and reset re-centres the active model and realigns the axes with the screen.



The transformation option pops up the view transformation panel. This allows rotation and translation of the model and saving of the current view or the loading of a view to and from the transmat.dat file.

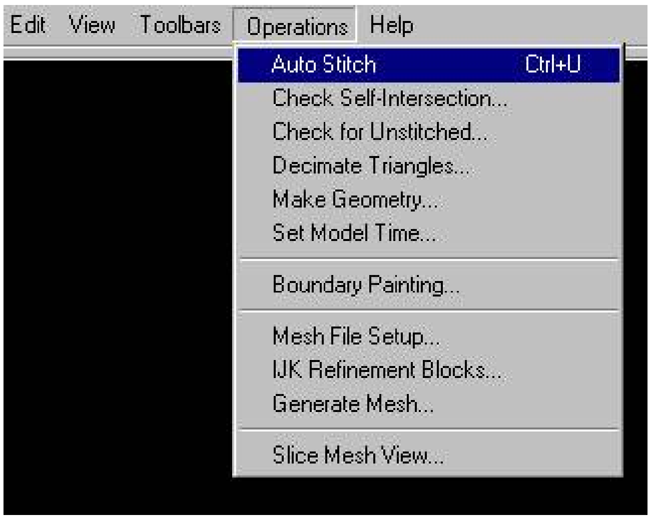


3.6 Triangle Processing

This section of the program is concerned with producing a triangulated surface model which meets the requirements of the VECTIS mesh generator.

The mesher requires that a surface model should be completely closed such that any point in space can be unambiguously determined as being inside or outside the model. This can only be achieved if every edge in the model lies on exactly two triangles. Equivalently, every triangle must be connected to exactly one other triangle on each of its three sides. Exceptions to these rules happen, for example, when a VDA is triangulated. Since a VDA file usually consists of many surfaces that are individually triangulated, the triangles on adjacent surfaces do not match, and thus require stitching together.

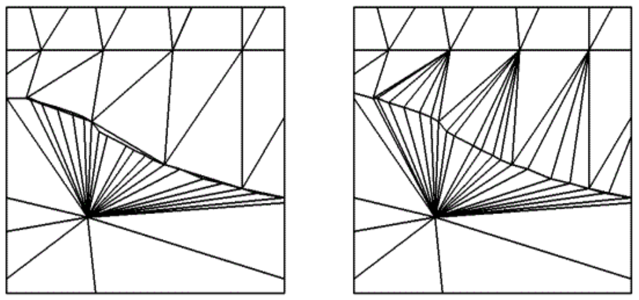
Some of this process may be completed automatically by the Phase 1 Auto Stitch command. This tool may be invoked by selecting the Auto Stitch option from the Operations menu.



or from the Auto-Stitch button

on the stitching tools panel.

An example of a section of geometry before and after stitching is shown in the Example Figure. It can be seen how the stitching inserts extra triangles in order to close the gaps between the surfaces. Stitching only operates on the active, visible part of the model (i.e. that part not removed by the Chop Area function).



Example of the Results of Auto-Stitching

Occasions may arise in which the introduction of additional triangles may result in spurious changes to the geometry. In such cases, the automatic stitcher makes cautious decisions to avoid introducing errors into the surface. Under these circumstances, it is unlikely that a fully stitched model will result and the remaining unstitched areas must be dealt with manually.

The tools used in this manual stitching process are available from the stitching toolbar. This toolbar will appear when any model file is opened by the GUI.

All of the operations available from this panel may be reversed by use of the Undo option from the 'Edit menu' or by the undo button

located on the button bar.

Phase 1 keeps a list of operations that can be reversed, discarding the list whenever something is done that can not be reversed. The Undo option is grayed out when there is nothing that can be undone. The space available to store the information necessary to undo operations is limited and therefore the number of steps that can be reversed is limited.

3.7 Triangle Creation Operations

Create Triangle Button

On selecting this operation, the cursor will change to a crosshair and instructions will be written to the information area. If three different nodes are selected by clicking upon them, and if these three nodes are valid vertices for a new triangle, a new triangle will be created using these nodes.

The only limit to the number of triangles that may be created, one after another, is the number of nodes available to create valid triangles with.

While choosing nodes to form the vertices of the new triangle, the nodes chosen will be marked with a white dot. If a required node is not visible in the current view of the model, the view may be adjusted with the mouse or keyboard as described above.

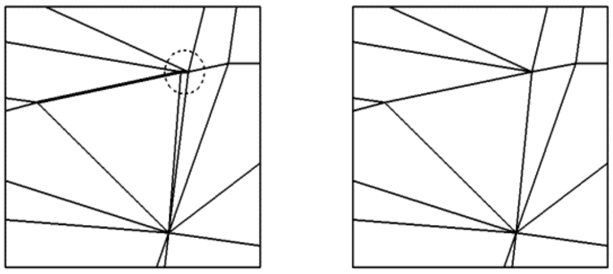
Delete Triangle Button

Once this operation has been selected, pressing the left mouse button with the crosshair cursor over a triangle will cause that triangle to be highlighted with a bold red fill. This triangle is now the deletion candidate. If the left button is clicked a second time over the same triangle, the triangle will then be deleted. If the escape key is pressed, or if a different triangle (or no triangle at all) is selected, the triangle deletion mode is exited and the Phase 1 returns to the "Manipulate View" state.

If a triangle is deleted, Phase 1 remains in the triangle deletion state and further triangles may be highlighted and deleted.

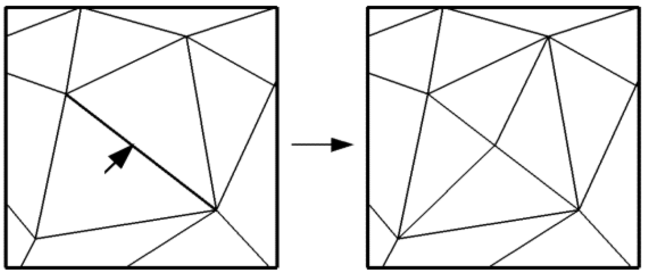
Join Triangles Button

This operation allows the user to merge a number of nodes together. The user may select the centre point of a circular region by clicking the left mouse button. Then the radius of the circle is defined by moving the mouse. And approximation of the circle remains bound to the mouse cursor while it moves. Clicking a second time fixes the size of the circle and performs the join operation. Any nodes inside the circle (and within a limited distance in the direction perpendicular to the screen) are merged together to give a single point with the average co-ordinate of all merged points. This function should be used with care. The user should only join points that are very close together, in order to avoid distortion of the model. An example of the use of this function is given in the Figure.


Joining Triangles to mend the surface

Split Line Button

This function will create a new node at the location selected upon an edge. The triangles sharing this edge will be re-triangulated to use this node.



Edge Splitting

Cap Hole Button

On selecting this operation, the cursor will change to a cross hair. If a line is then selected by clicking upon it, and the line forms part of a closed loop of a hole edge (drawn in red) in the surface, a number of triangles are created within the loop. Triangles are created between the lines describing the smallest angles first and stops once the hole is filled.

Auto Stitching Button

This is a direct way to do Auto-stitching it is an alternative to using the menu command Operations->Auto-stitch this button can be used to do the same operation. The user is prompted for a confirmation before the operation continues and the operation can be cancelled at this point if it is not required.

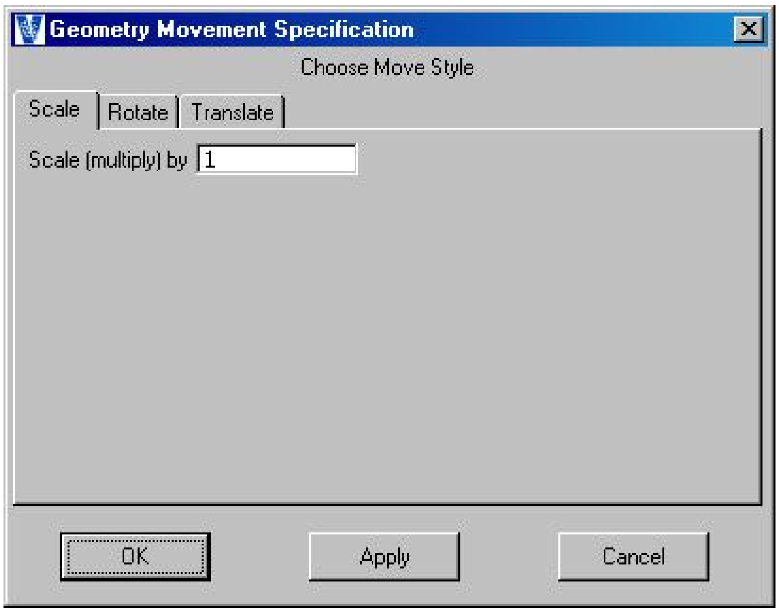
Flip Connected/Marked Triangles Button

This function reverses the normals of a connected set of triangles or a set defined with the mark triangles command. The triangle topology of the selected triangles is reversed and the normal is recalculated and will consequently point in the opposite direction.

Move Connected/Marked Area Button

This function works in conjunction with connected faces or the faces defined by the mark command. A face is selected by clicking with the left mouse button. A panel will then appear to allow the specification of a transformation to apply to the selected triangles.

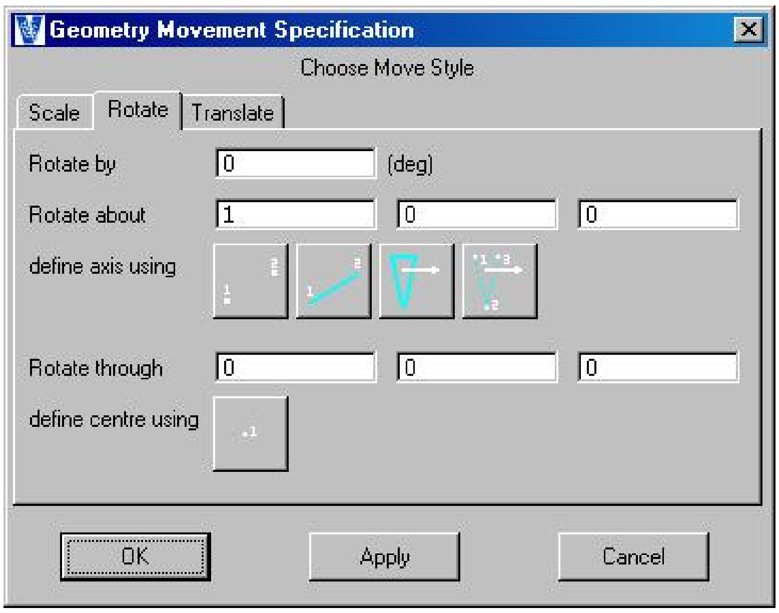
A scaling, rotation, or translation may be applied. For a scaling, only the scale factor needs to be supplied.



Scale (multiply) by 1

Scaling Transformation Panel

For a rotation, the angle of rotation (specified counter-clockwise), the axis vector to rotate about, and a point that the axis vector passes through, are needed. The rotation axis can be defined by entering the x,y,z vector or by selecting 2-points, 1-line, a triangle normal or a normal to 3-points. The point to rotate about can be entered or defined by selecting an existing point.


Geometry Movement Specification

Choose Move Style

Scale | Rotate | Translate

Rotate by 0 (deg)

Rotate about 1 0 0

define axis using

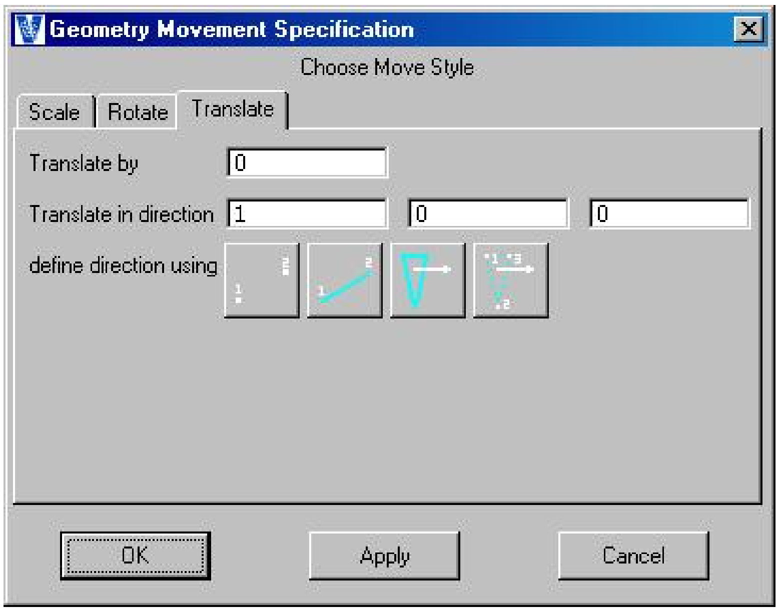
Rotate through 0 0 0

define centre using

OK Apply Cancel

Rotation Transformation Panel

For a translation, the distance and direction are specified. The direction vector for the transformation can be defined by entering the x,y,z vector or by selecting 2-points, 1-line, a triangle normal or a normal to 3-points.


Geometry Movement Specification

Choose Move Style

Scale | Rotate | Translate

Translate by 0

Translate in direction 1 0 0

define direction using

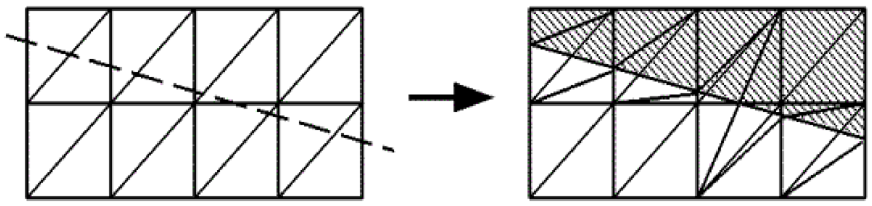
OK Apply Cancel

Translation Transformation Panel

3.8 Triangle Slicing and Intersection Operations

Horizontal Slice Button

When this operation is selected, a horizontal line becomes bound to the mouse pointer. When the left mouse button is clicked, the line thus defined is used to describe an infinite plane, perpendicular to the plane of the screen. This plane is used to "slice" through the visible (i.e. triangles which are not deactivated by the CHOP function) parts of the model, in such a way that new nodes are created on the slicing plane where the edges of the triangles intersect the plane. The triangles cut by the plane are divided into smaller triangles, and all triangles above the slicing plane are "marked." These marked triangles can be deleted (if desired), providing a convenient way of truncating geometry with a plane cut.


Triangle Slicing

If the model has been chopped, and then is saved, the user is asked whether to save the whole model, or just the visible (active) part. Thus, a partial model can be saved.

Vertical Slice Button

This function is as the horizontal chop function but with a vertical line.

Arbitrary Slice Button

When this function is selected, no line is bound to the cursor. Instead, a point on the canvas is selected and then a line is created from that point to the cursor. When the left button is clicked a second time, the line between the two selected points is used to create an infinite plane in the manner of the two previous functions.

Intersect Slice Selected Button

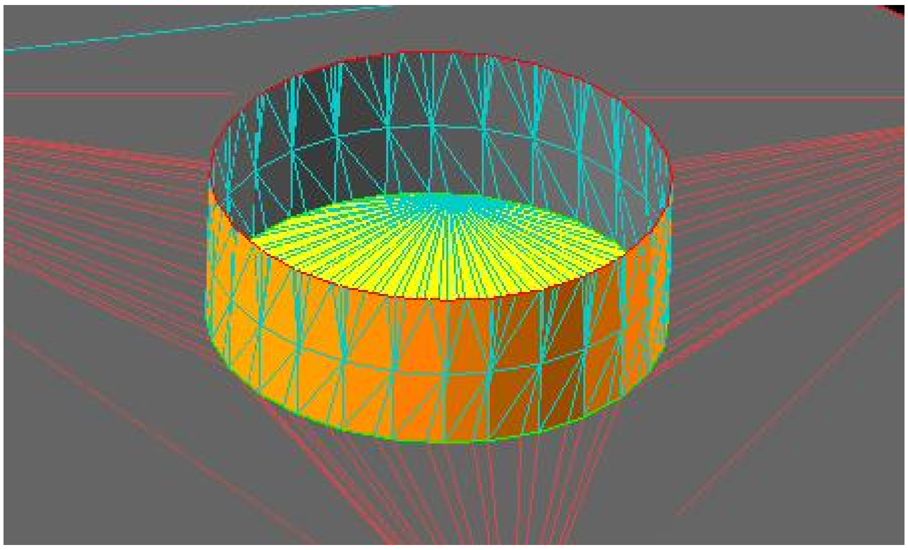
This function can be used to slice a selected surface along a line of intersecting triangles.

Intersect Slice Both Button

This function can be used to slice two intersecting surfaces along the line of intersecting triangles.

3.8.1 The Intersect Slice Operations

The functions currently require the user to have used the Check Self-Intersection function of phase1 to mark the triangles that intersect one another. This marks the intersecting triangles in red. If the user then wants to slice the intersecting triangles along the line of intersection the 'Intersect Slice selected' or 'Intersect Slice Both' buttons can be activated to slice the triangles of either the selected surface or both surfaces along the line of intersection respectively.



Slicing Along an Intersection

The program currently only calculates the intersections over a continuous area of marked intersecting geometry. If the program can't find an adjacent triangle that intersects it stops looking for other triangles to slice. This means several operations may be needed to slice the triangles in one boundary set.

The program calculates the points of intersection for each triangle and attempts to define a number of new triangles to replace the existing intersecting triangles. This is not always successful and is likely to fail if the geometric tolerances are small which may result in unstitched geometry being created. It is then necessary to stitch and repair the geometry where the program has failed to create triangles using the Auto-Stitch, CapHole and Create Triangle operations.

Once the new triangles have been created the program will attempt to mark the triangles on each side of the intersection line so the user then has to delete the unwanted geometry and stitch the remaining geometry.

If several operations are needed it might be necessary to do the check self-intersection operation a number of times to keep the intersecting triangles in the model correctly marked. This should not be too time consuming because the check self-intersection operation only checks active triangles. Triangles should not be coplanar.

The operation works better if the surfaces intersect 'fully' that is the surfaces do not intersect in a number of different places. In this case the check self-intersection routine will give a patchy result and a number of operations have to be done in succession to generate the intersection geometry.

3.9 Boolean Volume Operations

The user is able to perform Boolean type 'Volume Operations' on 'fully stitched closed volumes' in both GUI interactive mode and also batch mode. These are Union, Subtraction and Intersection operation involving at least two separate geometry entities.

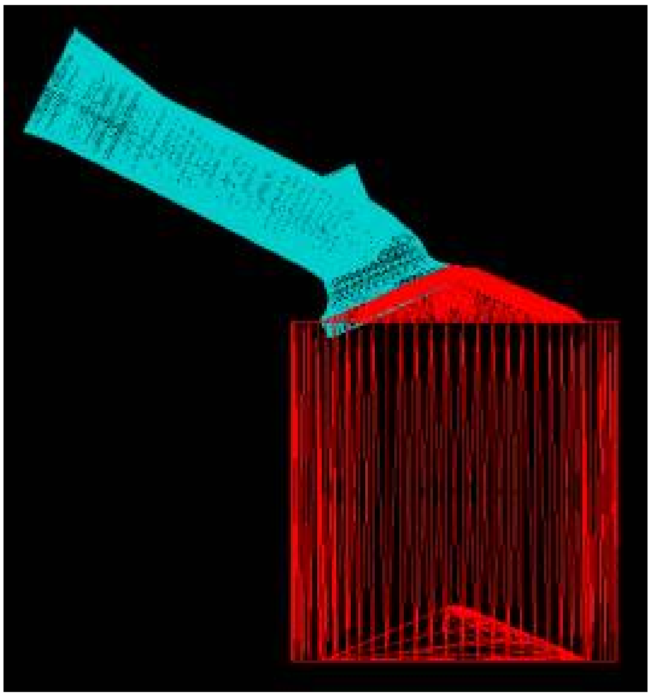
3.9.1 GUI interactive mode

The volume operation command buttons are:

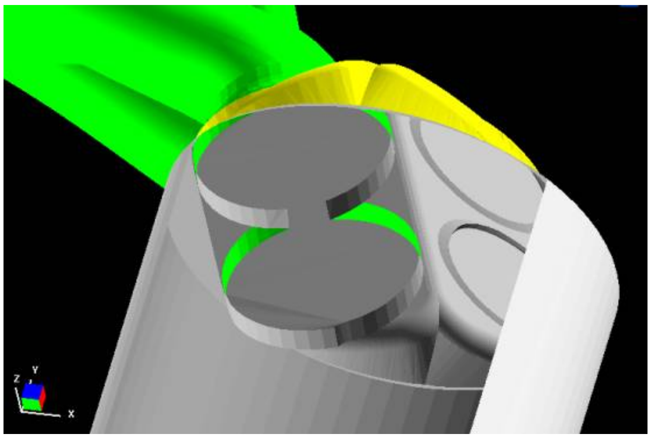
for 'Volume Union', 'Volume Subtract', and 'Volume Intersect' respectively.

When these command buttons are selected the user is prompted to select the volumes to operate on. A volume is selected by left clicking the mouse on a triangle that belongs to the required volume. The selected volume is then highlighted in red. Once at least two volumes have been selected the user is asked confirm the operation by firstly left clicking the mouse on the last volume selected and then choosing the OK button from the confirmation pop up window. More than two volumes can be selected by continuing to left click on different volumes so that the boolean operation once confirmed will then perform sequential operations using the volumes in the order they were selected.

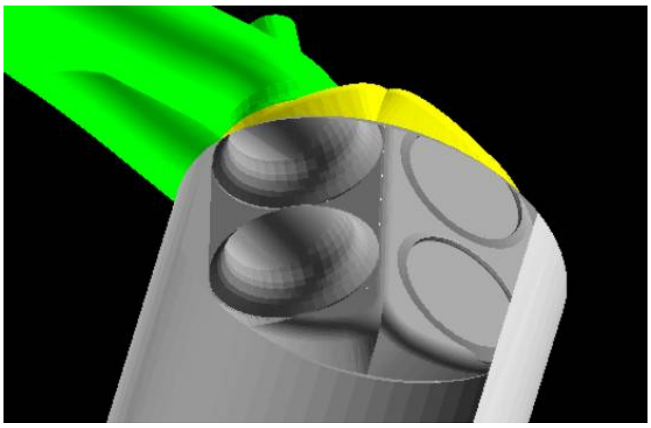
The Volume operations preserve boundary and part definitions so that the single resultant volume will be composed of the same parts and boundaries as the original volumes.



Cylinder and Port Volumes (red highlight)



Cutaway Showing Volume Intersection



Cutaway Showing Volume Union Result

3.9.2 Boolean Operation Batch Mode

The boolean operations are supported in batch mode in VECTIS version 3.9.0 onwards.

To run Phase1 in batch mode the command line arguments are:

```
phase1 [-bou | -bos | -boi] [-boundappend | -boundmerge]
       [-tolr tolrncy] filename1 filename2 filename_result
```

where:

-bou is the boolean operation UNION (default)

-bos is the boolean operation SUBTRACTION

-boi is the boolean operation INTERSECTION

-boundappend performs the boolean operation with the boundaries and parts of the second geometry appended after the boundaries and parts of the first geometry (default)

-boundmerge performs the boolean operation with the boundaries and parts of second geometry will be merged with the boundaries and parts of the first geometry.

-tolr tolrncy performs the boolean operation using the defined value for the merging tolerance of near nodes.

The default setting is AUTO which means that the program will automatically detect what the merge tolerance should be when loading the geometry files.

It is possible to use the batch mode capability in to perform consecutive boolean operations by simply having a script that performs the boolean operations where the file generated from the preceding operation is used as the input for the next operation. For example

```
phase1 -bou -boundaappend  A.tri    B.tri   AB.tri
phase1 -bou -boundaappend  AB.tri  C.tri   ABC.tri
```

3.10 Triangle Marking Operations

Mark Line Button

This operation is used to select a part of the model to which further commands can be applied.

The starting point is selected by clicking on the model with the left mouse button. A line will be drawn from the location clicked with the other end attached to the mouse. Further clicks of the left mouse button will create corners in the line.

Clicking twice in the same place will end the line.

Pressing the escape key or selecting another operation will cancel the line marking operation. Once the line has been drawn, all of the triangles cut by the line will be highlighted in a shaded red colour.

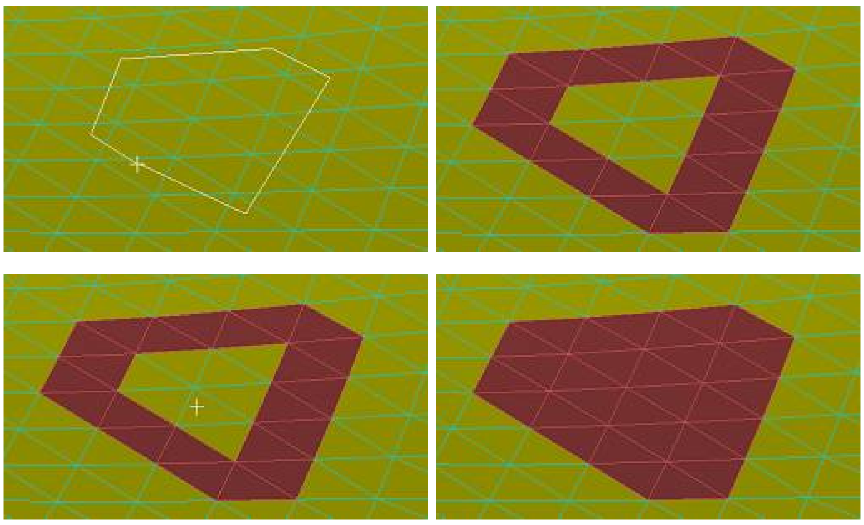
Once the operation has been completed, Phase 1 will return to the "Manipulate View" state. Note that the view may not be manipulated while the line is being drawn.

Mark Face Button

This operation is used to select a part of the model to which further commands will be applied.

A triangle is selected by clicking with the left mouse button and that triangle becomes the starting triangle for the construction of a face. The face contains all of the connected triangles going outwards from the first triangle until a boundary of marked triangles is met.

Therefore, to mark an area of the model, the perimeter of that area should be marked with the "Mark Line" command and then any triangle within that perimeter selected for the "Mark Face" command.


Example of Line and Face Marking

Mark Area Button

This operation is used to select a part of the model to which further commands will be applied.

Triangles are selected by describing the perimeter of a polygon which is drawn about the model by clicking the left mouse button. Two clicks in the same place will end the drawing process and close the polygon. The triangles inside the polygon will then be "marked" to which further commands can be applied.

Mark All Button

This operation is used to mark all of the model.

Delete Marked Area Button

This operation works on the faces defined using the marking functions described above. When the left button is clicked over a triangle, that triangle and all other triangles upon the same face are deleted.

Keep Marked Area Button

This function is the inverse of the previous one in that only triangles on the selected face remain - all other triangles are deleted.

Swap marked and Unmarked Button

This swaps over the faces that have been marked. All unmarked triangles become marked and the previously marked triangles are cleared.

Clear Marked Area Button

Pressing this button on the toolbar clears any marked faces from the model. The triangles return to their previous colour.

3.11 Triangle Chopping Operations

Chop Area Button

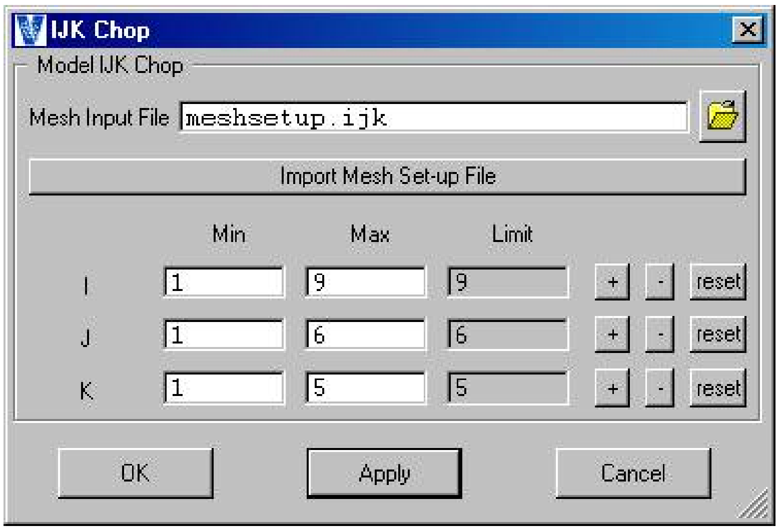
This operation is a useful feature for processing large models in that it enables the user to select a portion of a model for viewing, thus making the response time of dynamic manipulation faster.

The increase in manipulation speed gained by using Chop Area can be dramatic, and its use is recommended for all but the smallest of models.

The perimeter of a polygon may be drawn about the model by clicking the left mouse button. Two clicks in the same place will end the drawing process and close the polygon. The model will then be "chopped" so that only those triangles that lie completely inside the polygon are shown. The model can not be moved during chopping. Pressing the escape key during chopping cancels the function.

IJK Chop Button

Selecting this command pops up the ’IJK Chop’ panel.



Mesh Input File meshsetup.ijk

	Min	Max	Limit
I	1	9	9
J	1	6	6
K	1	5	5

This panel can be used to import an IJK block mesh set-up file and specify the IJK block extents to which the model should be chopped.

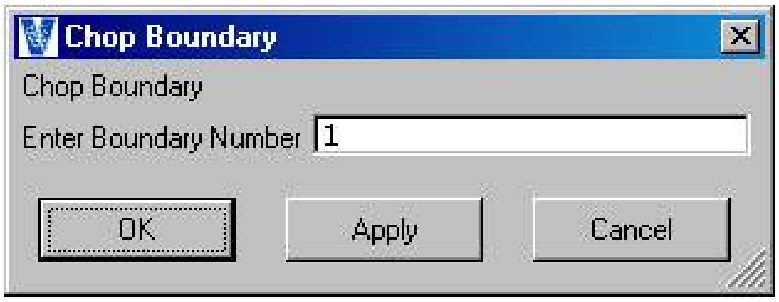
This makes all the triangles visible that have at least one node within the specified IJK block slice.

Chop Selected Boundary Button

This makes all the triangles invisible on the boundary that the selected triangle belongs to.

Chop Boundary by Number Button

Selecting this command pops up the ’Chop Boundary by Number’ panel



Chop Boundary

Enter Boundary Number 1

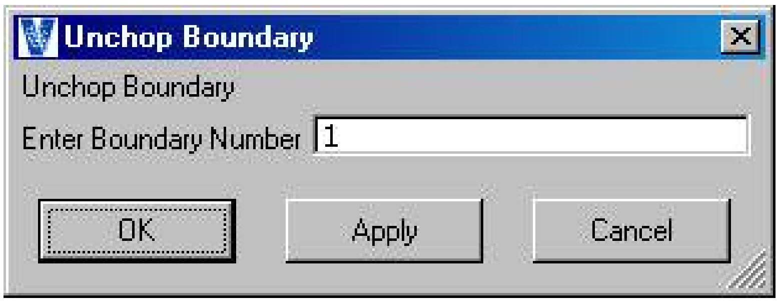
Specifying the boundary number and selecting OK makes all the triangles invisible on the specified numbered boundary.

Unchop Selected Boundary Button

This makes all the triangles visible on the boundary that the selected triangle belongs to.

Unchop Boundary by Number Button

Selecting this command pops up the ’Unchop Boundary by Number’ panel


Unchop Boundary

Unchop Boundary

Enter Boundary Number 1

Specifying the boundary number and selecting OK makes all the triangles visible on the specified numbered boundary.

Grow Active Set Button

This operation allows the user to grow the chopped model by a specified number of triangles.

A panel is popped up in which the user enters the number of triangles to grow the set by and inactive triangles adjacent to the active ones are made active up to the specified depth. Only triangles on active boundaries are made visible. Triangles on inactive boundaries remain invisible.

Grow Inactive Set Button

This operation allows the user to grow the chopped model by a specified number of triangles.

A panel is popped up in which the user enters the number of triangles to grow the set by and inactive triangles adjacent to the active ones are made active up to the specified depth. Triangles on both active and Inactive boundaries are made visible with this operation.

Swap Chopped and Unchopped Button

This swaps the visible and invisible portions of the model that lie on active boundaries created via the "Chop Area" command.

Show All Button

This makes all the triangles active and visible.

3.12 Triangle Interrogation Operations

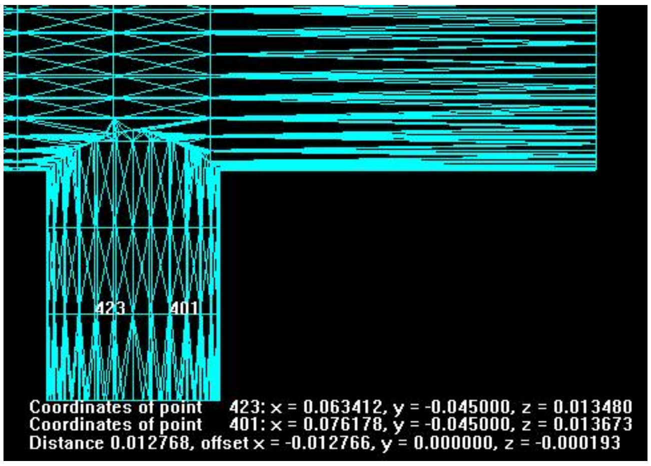
Co-ordinate display Button

While this mode is selected, the co-ordinates of the nodes that form the vertices of the triangles may be selected and their coordinate locations displayed.

Pressing the left mouse button near a node will display the number of that node and its 3d co-ordinates.

If another node is selected by single clicking the left mouse button the spatial distances between the nodes are printed for the three co-ordinate directions as well as the direct path length distance.

If another node is selected by double clicking the left mouse button then the cumulative distances and path lengths are printed and this can be repeated for many nodes.

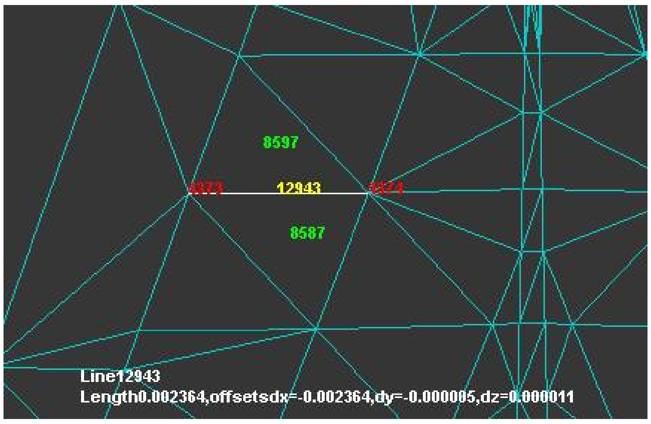


```
Coordinates of point     423: x = 0.063412, y = -0.045000, z = 0.013480
Coordinates of point     401: x = 0.076178, y = -0.045000, z = 0.013673
Distance 0.012768, offset x = -0.012766, y = 0.000000, z = -0.000193
```

Node co-ordinate display

Interrogate Line Button

Once this command has been selected the user can select a line on the model and the line number will be displayed. In addition the line end nodes and the line's adjacent triangle numbers are displayed together with the line length and the direction vector the line defines.



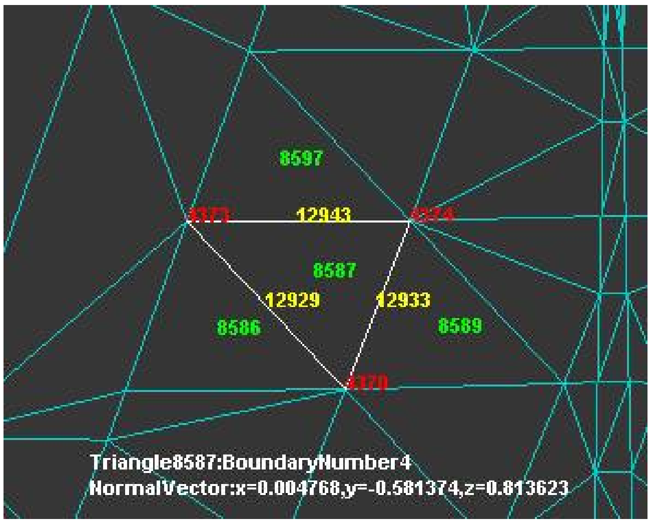
```
Line12943
Length0.002364,offsetsdx=-0.002364,dy=-0.000005,dz=0.000011
```

Interrogate Line display

Interrogate Triangle Button

Once this command has been selected the user can select a triangle on the model and the triangle number will be displayed.

In addition the triangle's line numbers, line end nodes and the triangle's adjacent triangle numbers are displayed together with the triangles boundary number and the triangle's normal vector components.



```
Triangle8587:BoundaryNumber4
NormalVector:x=0.004768,y=-0.581374,z=0.813623
```

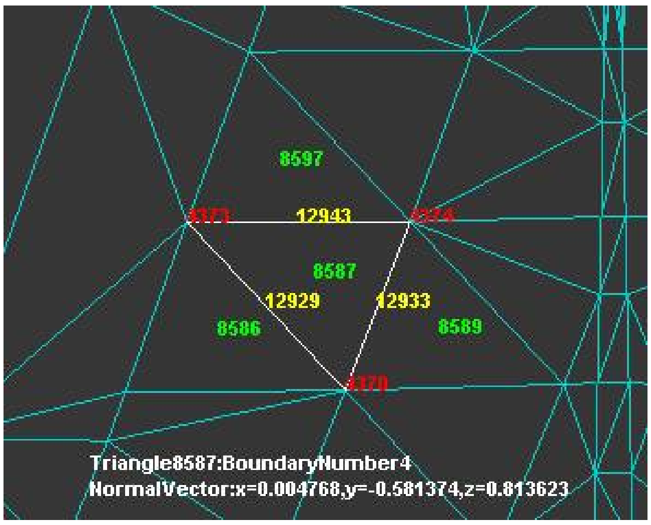
Interrogate Triangle display

Interrogate Boundary/Marked Region Button

Once this command has been selected the user can select a triangle in the model and the triangle number will be displayed.

In addition the triangle's boundary number, the surface area of the boundary region and the centroid of the boundary region are displayed.

The user can either interrogate a boundary region or a region of triangles which have been marked with the marking commands.



```
Triangle8587:BoundaryNumber4
NormalVector:x=0.004768,y=-0.581374,z=0.813623
```

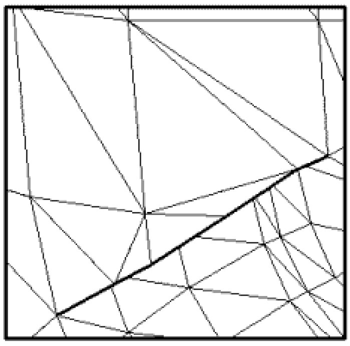
Interrogate Boundary/Marked Region display

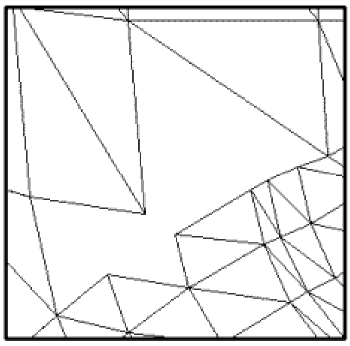
3.13 Hints for Manual Stitching

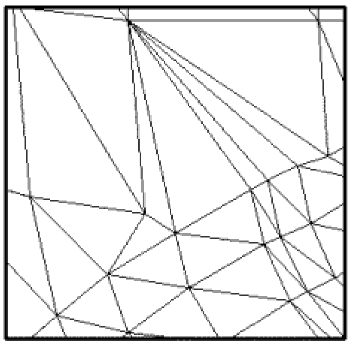
There is usually more than one possible way to use Delete Triangle, Create Triangle, and Join Triangles operations in a given situation. The user will find that the "correct" form of the surface is very often clear. The way to approach an area that needs stitching is usually to delete triangles that are wrong, and then to fill in the gap created with new triangles.

Another feature of the surface which should be checked once stitching is complete, is that the model does not contain overlapping, or tightly folded surfaces. To allow the identification of such regions, any edges in the model which lie on two triangles whose normals have an angle of more than 170 degrees between them are highlighted in green at all times.

It is good practice to look for these regions by turning off the display of surfaces and outlines. There may be very sharp edges in the geometry which are quite correct, but there may also be areas where triangles lie on top of each other, or where the surface is folded back on itself. These may become a source of ambiguity in the surface for the mesher, and they should be eliminated with the use of the manual stitching operations.


The bold line marks a set of unstitched triangle edges. The triangles to the upper left overlap those to the lower right.


The overlapping triangles are deleted.


New triangles are created to fill in the gaps.

3.13.1 Triangle Reduction

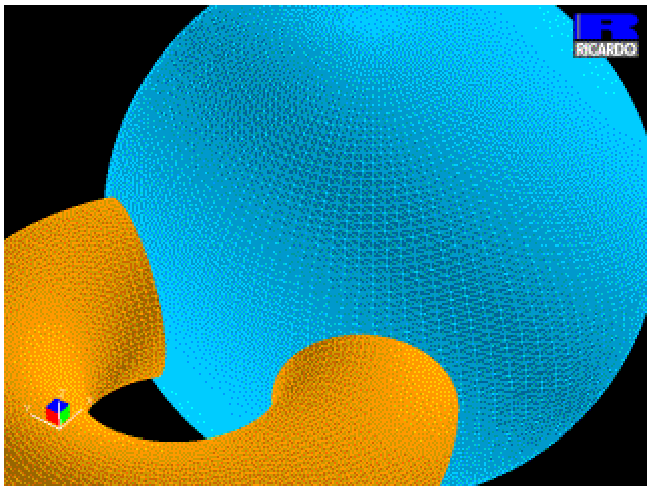
In some cases, it may be desirable to reduce the number of triangles in a model by removing unnecessary detail. This can help speed up manipulation of the model on the screen, as well as saving memory and runtime in subsequent phases of VECTIS.

Selecting Decimate Triangles from the Operations menu performs this operation. This function works by collapsing short edges in the model to a single point. The user is prompted to enter the limiting triangle edge length below which collapses should be performed. Each line collapse operation removes two triangles. The algorithm is constrained so that it does not alter the topology of the model, and so that sharp edges are preserved. The triangle reduction function operates only on the active part of the model.

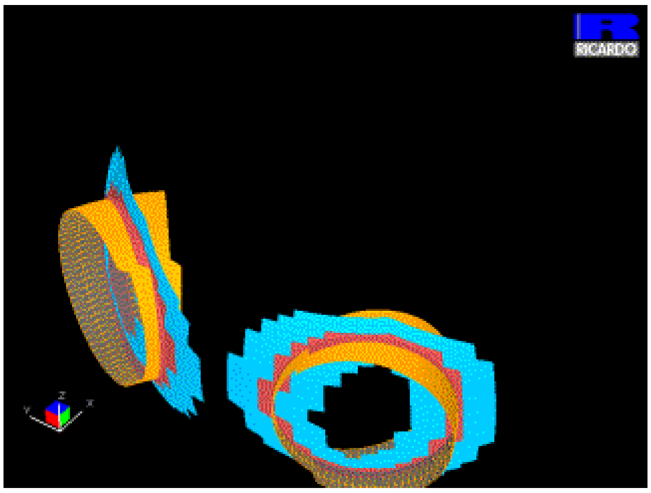
3.13.2 Self Intersection Checking

This function performs checks to see if any of the triangles in the currently active part of the model intersect each other. It is useful to identify bad geometry which may not be handled properly in the mesher.

The function is selected from the operations menu. This brings up a confirmation panel, with a field in which to enter the intersection border depth (the meaning of which is described below). Clicking on OK starts the intersection check, which may take some time for a large model. A progress indicator shows how many triangles have been processed. An example of the operation of the function is illustrated in the following figures, which show a model consisting of an intersecting torus and sphere.



Sphere and torus before intersection check



After the intersection check, the intersecting triangles are highlighted in red. All other triangles are deactivated, except for a "context" border which allows the user to recognize the self intersecting sections more easily. The depth of this border is controlled by the intersection border depth entered before performing the check.

3.14 Geometry Wrapping

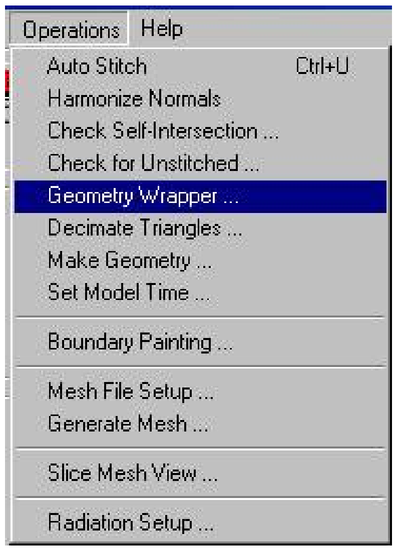
The VECTIS Phase 1 geometry wrapper generates an external triangular surface mesh around an existing geometry model, simplifying the details and removing internal triangles. It can be used as a clean-up tool for reasonable-size geometries. However, the user should be aware that deviations from the initial geometries can occur. The wrapping process is controlled by several input parameters. A new surface mesh is generated automatically however some manual repair can be required.

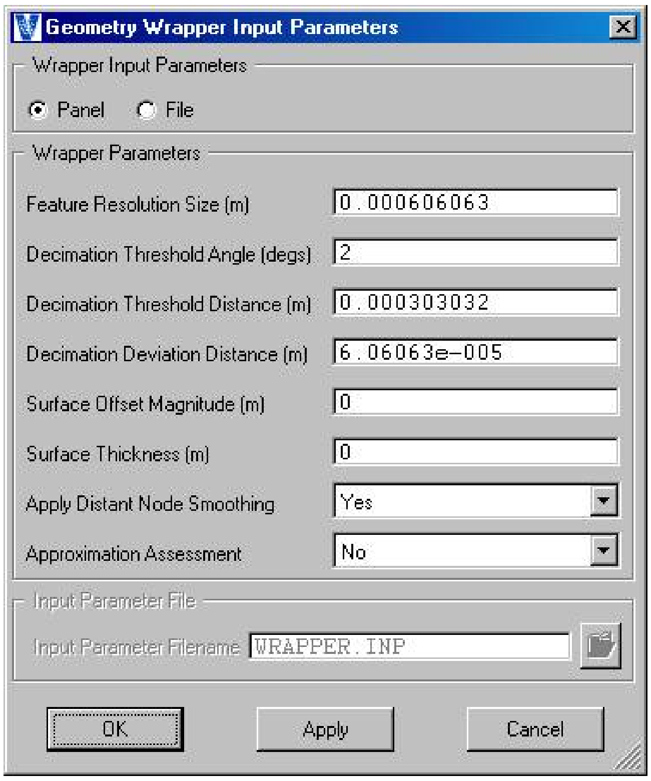
The wrapping algorithm is based on projection mesh generation methods. The input geometry is placed in a cubic box (domain). The boundaries of the geometry are surrounded by the Cartesian cells (base mesh) having no intersections with the geometry. External cell faces of the base mesh are projected onto the boundaries. And after that the surface triangles are decimated in order to reduce their number. Using this technique small holes in the initial geometry, over-lapping, self-crossings are allowed and geometry triangular connectivity is not important.

The geometry wrapper can be used in the GUI interactive mode or in a batch mode with the wrapper settings defined in the GUI or from a parameter file. NOTE that some features may only be available by using the parameter input file method since the GUI may not support them yet. For example in VECTIS 3.9.0 the leak detection mode can only be used with the input file option.

3.14.1 GUI Interactive Mode

The Wrapper is invoked by selecting the phase1 Operations > Geometry Wrapper... menu item from the which displays the wrapper input parameter panel.




Geometry Wrapper Input Parameters

Wrapper Input Parameters

[✓ Panel] File

Wrapper Parameters

Feature Resolution Size (m)	0.000606063
Decimation Threshold Angle (degs)	2
Decimation Threshold Distance (m)	0.000303032
Decimation Deviation Distance (m)	6.06063e-005
Surface Offset Magnitude (m)	0
Surface Thickness (m)	0
Apply Distant Node Smoothing	Yes
Approximation Assessment	No

Input Parameter File

Input Parameter Filename WRAPPER.INP

Geometry Wrapper Input Panel

Wrapper input parameters can be specified either in the panel or can be read from a file by selecting the ’File’ radio button. When the ’File’ radio button is selected the wrapper parameter input filename can be specified using the input parameter file browser. Refer to the Geometry Wrapper Input File section and Appendix D for more information regarding the wrapper parameter file.

The main input parameter is the ’Feature Resolution Size’. It corresponds to the size of minimal base mesh surrounding the initial geometry. The lower this parameter, the higher the resolution that can be achieved. The default value in the menu corresponds to 128 divisions or smallest Cartesian cells in the largest direction of the model.

The initial geometry can be any set of triangles. To simplify the geometry correctly, holes in the initial geometry should be no larger than ’Feature Resolution Size’. Larger holes may cause leaks of the Cartesian mesh inside the geometry. The result of leaks can be easily visible in Phase 1 as a mixture of triangles with both sides. The running time also increases. It is suggested to carefully inspect the initial geometry and cap all relatively large holes, start the Wrapper with relatively large ’Feature Resolution Size’ and gradually reduce it.

Triangle decimation is on by default. The ’Decimation Threshold Angle’ is based on the angle between adjacent triangle normals. Large values of the ’Decimation Threshold Angle’ should not be applied to thin geometries, since it can lead to self-intersections. The default value 2 degrees usually gives satisfactory results.

Triangle decimation is also based on edge collapsing, but for user-specified edge length and a high angle tolerance. It is recommended to set the decimation distance to be not larger than half the feature resolution size and use a combination of the decimation based on angle and the decimation based on distance.

The deviation decimation edge-collapsing technique uses deviation criterion from the wrapped model to the original geometry. It can produce more accurate results, but requires more time. The recommended value for the deviation distance is one tenth of the feature resolution size. It is also suggested to use the decimation based on angle and the decimation based on distance prior to using the deviation-based decimation.

The 'Surface Offset Magnitude' feature enables an offset to be applied to the wrapped geometry. The offset is based on moving the triangles along their normal direction by a defined distance. If the input value is zero, no offset occurs. In case of the positive value, the wrapped surfaces grows. If the offset distance is negative, it shrinks. The allowed range is [-fr, +fr]. To obtain a larger offset, the wrapping process with the offset option should be repeated several times. It is recommended to use the decimation angle and decimation distance option for element reduction when using the offset option.

In the case of leak occurring during a wrapping process the surface offset can be used for quick hole capping. The following operations are suggested:

Wrap the geometry with a coarse fr size without leaks applying a small negative offset distance.

Merge the wrapped geometry with the original geometry.

Repeat the wrapping process on the combined original and coarsely wrapped shrunk surface with a fine feature resolution size.

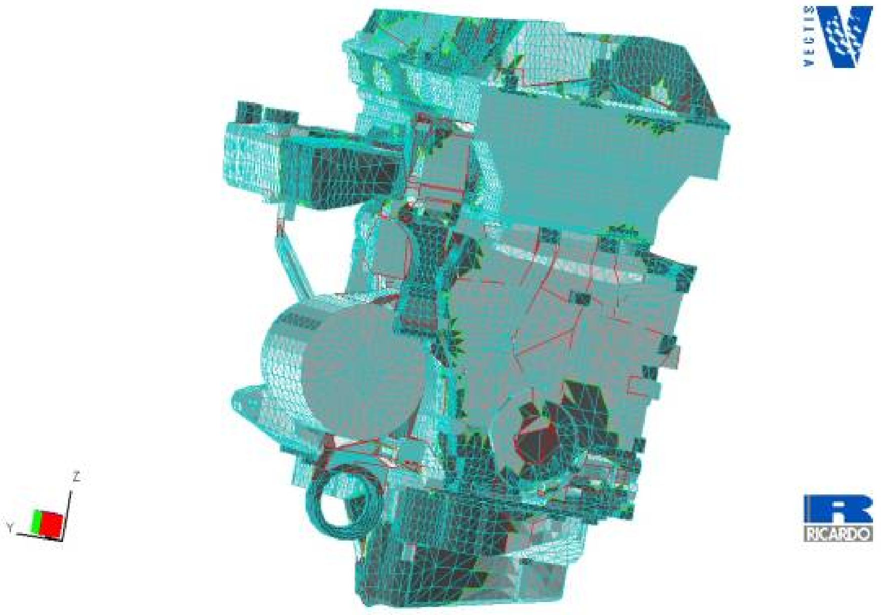
The 'Surface Thickness' option can be used to wrap infinitely thin surfaces i.e. shells. It works by creating a thickness for infinitely thin surfaces. For more accurate and faster results it is strongly recommended to apply the surface thickening option to the parts of the original model that are infinitely thin as a separate wrapping process, then merge the thickened parts with the original geometry and run the Wrapper without the surface thickening mode.

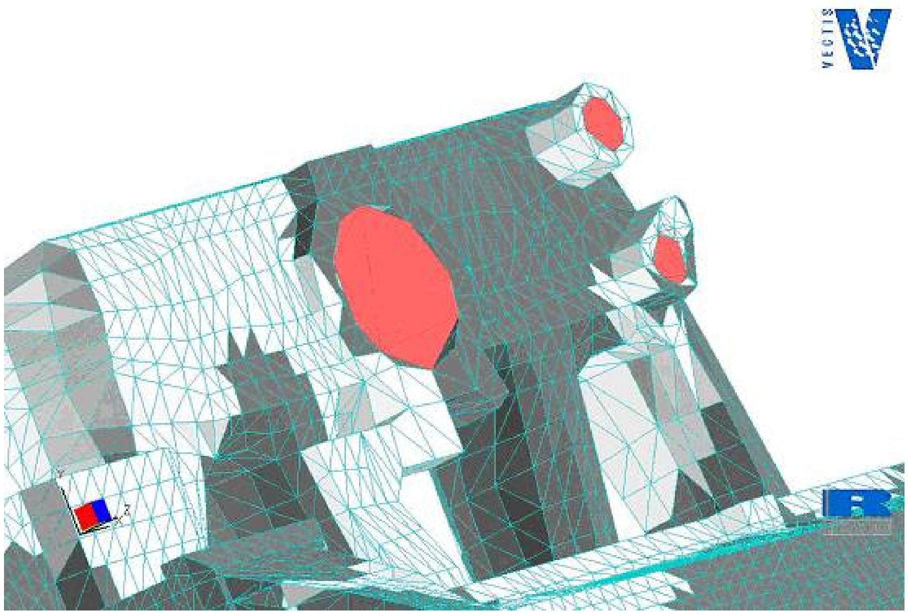
To reduce artificial mesh disturbances near concave or poorly resolved features nodes located relatively far from the original geometry can be smoothed using 'Distant Node Smoothing' mode. This option can be recommended for the majority of geometries.

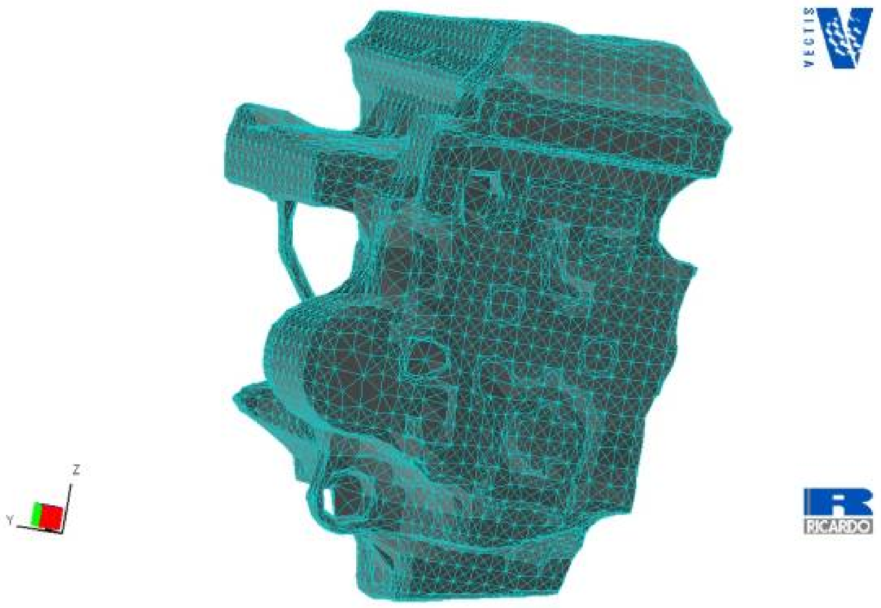
The quality of wrapper approximation can be assessed using distances from triangles to the original geometry. Using this mode would take slightly more time. The output data is appended into a file called 'Deviation.rp' which can be opened in the Ricardo plotting program RPLOT and displays a distance vs percentage of area of triangles further from the geometry than this distance. The approximation assessment is not available in when using the leak detection mode through the wrapper input file.

The geometry wrapper returns the wrapping time, number of output triangles and wrapped surface area. When using the approximation assessment mode, the minimum, maximum and average distances from triangles to the original geometry is also reported. The latter is not available when using the surface thickening and

The output information is displayed on the screen and is written into file 'PHASE1.OUT'. Note, that the new information is appended.

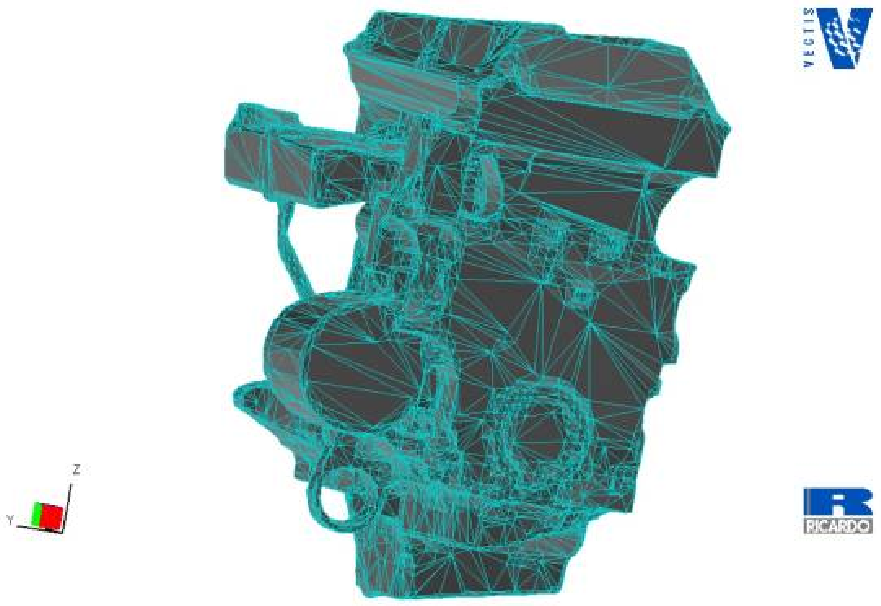

Raw Engine Geometry


Rough capping of Large Holes/Features



Example of Wrapping

Feature resolution size 0.02m, no maximize feature resolution, no decimation.



Example of Wrapping

Feature resolution size 0.01m, maximize feature resolution, threshold angle 5 degrees.

3.14.2 Batch Mode

The geometry wrapper can be run in batch mode.

The batch mode wrapping command line arguments are:

```
phase1 -w [-fr feature_resolution]
[-ld leak_detection_mode]
[-da decimation_angle]
[-dd decimation_distance]
[-dv deviation_distance]
[-od offset_distance]
[-st surface_thckness]
[-ds distant_node_smoothing]
[-aa approximation_assessment]
[-i input_parameters_filename]
[-o output_filename]
input_tri_filename
```

A default fr size is applied if no value is specified. A default da is 2 degrees. By default ds is applied. If no parameter is specified in command line, it will be read from the parameters file. Boundary fr sizes and maximum refinement boundaries are read from the parameters file. The no output filename is specified, the file extension .wrp is used for the output filename.

Feature resolution size

This parameter approximately corresponds to the smallest features of the original geometry that will be automatically recovered. If the geometry contains close surfaces, fr size should be 1/3 of the smallest distance between them. The algorithm is optimised for the fr values printed onto screen/menu.

Boundary feature resolution size

It is possible to set fr sizes for boundaries. This is currently available only in the Wrapper parameters file using keyword `BOUNDARY_FEATURE_RESOLUTION_SIZE`. Boundary fr sizes can be smaller than the global fr size. Base mesh will be refined to the level not exceeding the specified sizes. This can enhance resolution of particular features, for example close surfaces.

Maximum refinement

Instead of boundary fr sizes it is possible to set certain boundaries refined to the maximum selected fr size. This also can be done only in the Wrapper parameters file using the following key word: `MAXIMUM_REFINEMENT_BOUNDARY`.

Leak detection

In case of base mesh 'leaks' inside, the geometry will be wrapped from inside and outside. In this case usually memory and CPU usage are significantly increased. One of the ways to prevent this is to cap 'large' holes in the original geometry. The Wrapper can be run in leak detection mode generating WrapperLeakPaths.stl for visualisation of leak paths. This mode runs faster and does not require some parameters. Leak paths connect external points with user-specified internal ones. The maximum number of internal points is 24. If an internal node is too close to the geometry a warning will be printed and the closest internal point will be selected. For internal points setting it is recommended to wrap the geometry, use Phase1 'Chop Area', 'Create triangle', 'Split Line' and 'Measure Node' tools.

Decimation angle

To reduce the number of output triangles angle-based decimation can be used. It collapses edge, preserving normal angles between old and new triangles at user-specified interval. Recommended value for the threshold is 2-3 degrees. The maximum allowed da is 10 degrees. Please note that for thin geometries self-intersection can occur after decimation.

Decimation distance

This decimation is also based on edge collapsing, but for user-specified edge length and a high angle tolerance. It is recommended to set dd not larger than fr/2 and use a combination of da and dd.

Deviation distance

This edge-collapsing technique uses deviation criterion from the wrapped model to the original geometry. It can produce more accurate results, but requires more time. The recommended value for deviation distance (dv) is fr/10. It is also suggested to use da and dd prior deviation-based decimation.

Offset distance

This feature enables an offset to be applied to the wrapped geometry. The offset is based on moving the triangles along their normal direction by a defined distance. If the input value is zero, no offset occurs. In case of the positive value, the wrapped surfaces grows. If the offset distance is negative, it shrinks. The allowed range is [-fr, +fr]. To obtain a larger offset, the wrapping process with the offset option should be repeated several times. It is recommended to use da and dd for element reduction with offset option.

In case of leak, surface offset can be used for quick hole capping. The following operations are suggested:

1. Wrap the geometry with a coarse fr size without leaks applying a small negative offset distance.
2. Merge the wrapped geometry with the original geometry.
3. Repeat the wrapping process on the combined original and coarsely wrapped shrunk surface with a fine feature resolution size.

Surface thickness

This option produces more accurate positive surface offset up to fr size for thin geometries, but requires more CPU resources. It can be used to thicken infinitely thin surfaces. For more accurate and faster results it is strongly recommended to apply st to some parts of the original model, merge thickened parts with the original geometry and run the Wrapper without st mode.

Distant node smoothing

To reduce artificial mesh disturbances near concave or poorly resolved features nodes located relatively far from the original geometry can be smoothed using distant node smoothing mode. This option can be recommended for the majority of geometries.

Approximation assessment

The quality of approximation can be assessed using distances from triangles to the original geometry. Using this mode would take slightly more time. The output data is appended into 'Deviation.rp' file providing distance vs percentage of area of triangles further from the geometry than this distance. aa is not available in ld mode.

Statistics

Geometry Wrapper returns the wrapping time, number of output triangles and wrapped surface area. In case of aa mode, minimum, maximum and average distances from triangles to the original geometry is also reported. The latter is not available in st and ld modes.

Output file

The output information about progress and results is reported into Phase1 menu and appended to PHASE1.OUT.

3.14.3 Geometry Wrapper Input File

A file browser in 'Geometry Wrapper Panel' allows the user to specify the use of wrapper parameters input file of the format as shown by the example below.

The definitions in this file will be used instead of the GUI panel input values. Refer to Appendix D for more information regarding the wrapper parameter file.

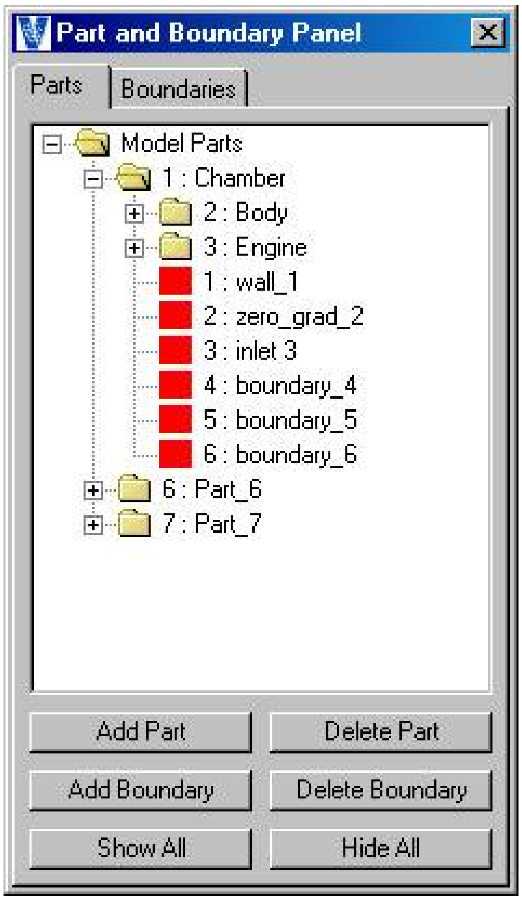
```
!VECTIS_WRAPPER_INPUT VERSION 3.9
#====================================
FEATURE_RESOLUTION_SIZE
```

```
0.003
#=======================================
MAXIMUM_REFINEMENT_BOUNDARY
1 4 5
#=======================================
BOUNDARY_FEATURE_RESOLUTION_SIZE
2 0.001
#=======================================
BOUNDARY_FEATURE_RESOLUTION_SIZE
3 0.0005
#=======================================
DISTANT_NODE_SMOOTHING
T
#=======================================
DECIMATION_ANGLE
1.
#=======================================
DECIMATION_DISTANCE
0.001
#=======================================
DEVIATION_DISTANCE
0.0003
#=======================================
SURFACE_THICKNESS
0.
#=======================================
OFFSET_DISTANCE
0.
#=======================================
APPROXIMATION_ASSESSMENT
T
#=======================================
LEAK_DETECTION_MODE
F
#=======================================
INTERNAL_POINT
0.115518 -0.007180 0.000471
#=======================================
INTERNAL_POINT
0.033556 0.038563 -0.002235
#=======================================
INTERNAL_POINT
0.074374 -0.045267 0.003891
#=======================================
```

3.15 Boundary Processing

3.15.1 Part and Boundary Definition

In Phase1 it is possible to split a model up into parts. This facilitates model manipulation and part substitution.

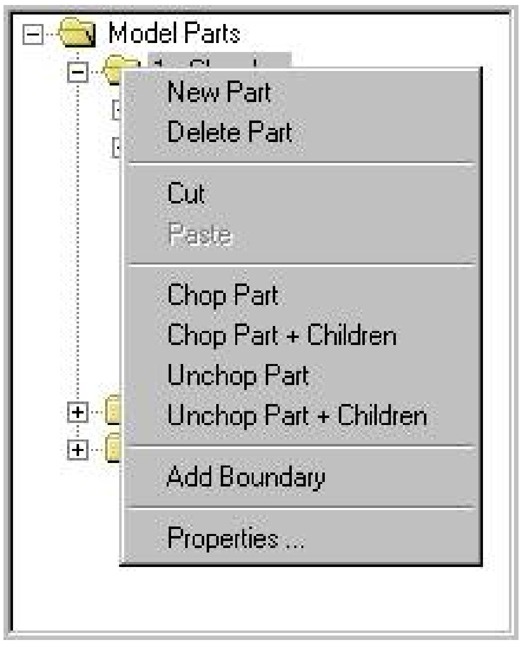


Part Panel

The part tree displays the nested part structure of the current model.

Part pop-up menu operations

Selecting a part with the Right Mouse Button causes the part pop-up menu to be displayed.



Part pop-up menu

The Part Pop-up menu allows the user to add and delete parts, cut and paste parts into other parts, chop and unchop parts and their children, add new boundaries and set part properties.

Add/Delete Part

The ’Add Part’ function adds a part definition without assigning boundaries or triangles to it and assigns it a default part name in the form `part_#`. If the ’Delete Part’ function is selected the user is prompted to delete both the part and the triangles in the part or to delete just the triangles in the selected part.

Cut/Paste Part

Selecting ’Cut’ selects a part to cut which can subsequently be pasted into another part using the part pop-up menu. The ’Paste’ menu item becomes selectable after either a boundary or a part has been cut. The cut item (Boundary or Part) can then be pasted into a part using the ’Paste’ operation.

Chop/Unchop Part

Selecting Chop Part and Unchop Part makes the part Active or Inactive respectively. Users can optionally chop and unchop the children of parts and make whole branches of the model active or inactive in one operation.

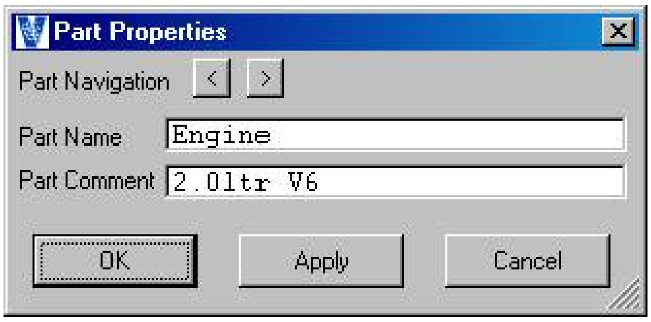
Add Boundary

Adds a new boundary to the current part with no triangles assigned to it.

Part Properties

Selecting ’Part Properties’ pops up the ’Part Properties Panel’ which can be used to change part names and add or edit comments for the parts.

Part Properties Panel


Part Properties

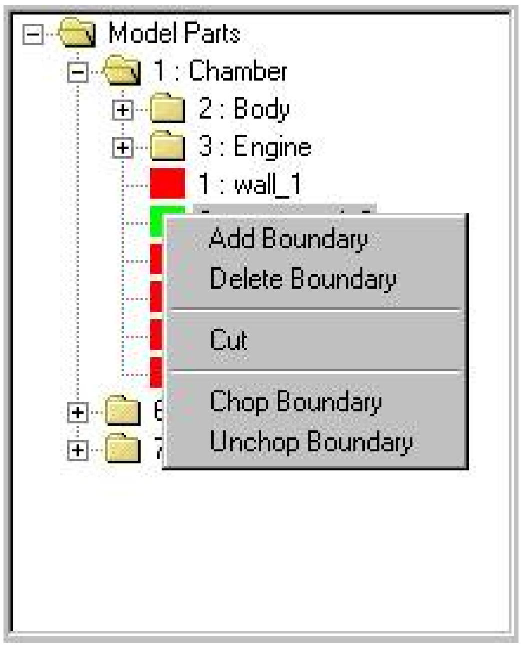
Part Navigation: < >

Part Name: Engine

Part Comment: 2.0ltr V6

Part properties menu

Selecting a part with the Right Mouse Button causes the part pop-up menu to be displayed.



Boundary pop-up menu

The Boundary Pop-up menu allows the user to add and delete boundaries, cut and paste boundaries into other parts and chop and unchop boundaries.

Add/Delete Boundary

The 'Add Boundary' function adds a boundary definition without assigning triangles to it and assigns it a default boundary name in the form `boundary_#`. If the 'Delete Boundary' function is selected the user is prompted to delete both the boundary and the triangles on the boundary or to delete just the triangles on the selected boundary.

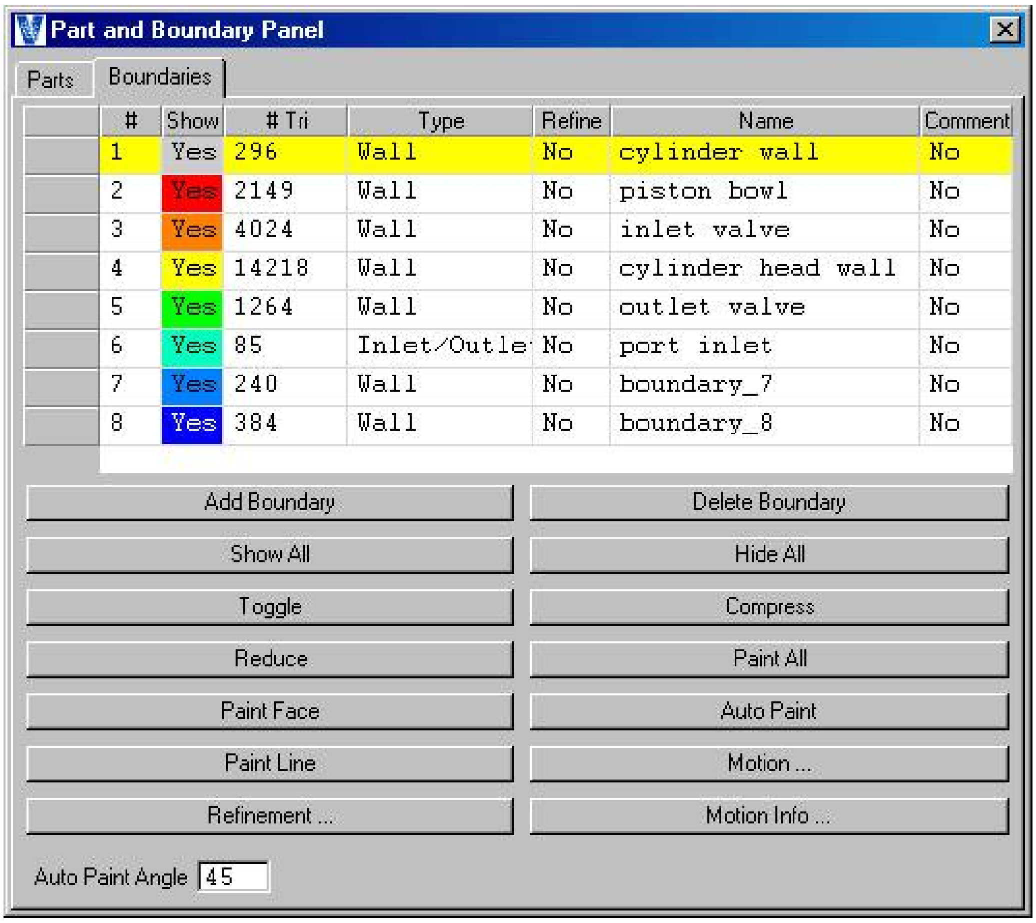
Cut Boundary

Selecting 'Cut' selects a boundary to cut which can subsequently be pasted into another part using the part pop-up menu.

Chop/Unchop Boundary

Selecting Chop Boundary and Unchop Boundary makes the boundary Active or Inactive respectively and is the same as switching a boundary on or off by using the 'Show' column in the boundary panel. An inactive boundary is not displayed on the screen and so boundaries may be quickly removed to gain an unobstructed view of hidden parts of the model.

3.15.2 Boundary Processing


Part and Boundary Panel

Parts | Boundaries

#	Show	# Tri	Type	Refine	Name	Comment
1	Yes	296	Wall	No	cylinder wall	No
2	Yes	2149	Wall	No	piston bowl	No
3	Yes	4024	Wall	No	inlet valve	No
4	Yes	14218	Wall	No	cylinder head wall	No
5	Yes	1264	Wall	No	outlet valve	No
6	Yes	85	Inlet/Outle	No	port inlet	No
7	Yes	240	Wall	No	boundary_7	No
8	Yes	384	Wall	No	boundary_8	No

Auto Paint Angle 45

Before a calculation may be run on a model it is necessary to define the boundaries of the model. For example, when calculating the flow in the inlet port and cylinder of an engine, the user may define the piston crown as one boundary, the cylinder liner as another, the valve surface as another, the port inlet plane another etc. A panel of tools is provided to set up this information.

The table as shown to the left displays information related to each boundary.

The first column simply holds a unique identifier number for that boundary.

The second column is filled with the colour that triangles on that boundary are filled with. By clicking the left mouse button on a cell in that column, the text in the cell will alternate between Active and Inactive. An inactive boundary is not displayed on the screen and so boundaries may be quickly removed to gain an unobstructed view of hidden parts of the model.

The third column shows the number of triangles currently assigned to that boundary. By left clicking and holding the mouse button over a cell in this column and dragging the mouse pointer to another cell in this column, triangles may be moved from boundary to boundary.

The type of the boundary as far as the calculation is concerned is set in the fourth column. Clicking the left mouse button in the cell cycles between the options (Wall, Zero Grad, Inlet/Outlet, and Cyclic). Boundaries may be given unique names by typing them into the sixth column.

The fifth column indicates whether the boundary has refinement defined for it. Selecting the table cell pops up the boundary refinement panel in which the boundary refinement is defined. Then selecting Ok pops down the panel and the cell displays 'Yes' to indicate that boundary refinement has been defined.

The sixth column is used to define a boundary name for each universal boundary number. Each boundary name can be up to 80 characters long.

Comments can be defined for each boundary. The seventh column indicates whether the boundary has a comment defined for it. Selecting the table cell pops up the boundary comment panel in which the boundary comment can be set. Then selecting Ok pops down the panel and the cell displays 'Yes' to indicate that boundary comment exists.

The highlighting upon one row of the table indicates the currently selected boundary. This may be changed by clicking the left mouse button in the identifier number column for the boundary to be selected. This selection determines which boundary triangles are added to by the boundary painting functions described below.

Add/Delete Boundary

The 'Add boundary' function adds a boundary definition without assigning triangles to it. The 'Delete Boundary' function deletes the triangles on the selected boundary and sets its type to wall.

Show/Hide All Boundaries

The 'Show All' function switches all the boundaries to active and displays them if they have not been chopped and the 'Hide All' function switches all the boundaries to inactive so that the boundaries will no longer be displayed.

Toggle Boundaries

The 'Toggle' function switches all the active boundaries to inactive and so they will no longer be displayed and switches all the inactive boundaries to active and displays them if they have not been chopped.

Compress Boundaries

The 'Compress' function removes all boundary definitions which have no triangles assigned to them.

Boundary Painting

The Paint Line and Paint Face buttons operate in much the same way as the Mark Line and Mark Face stitching operations. Instead of marking the triangles however, these functions change the boundary that the triangles are assigned to. This results in the colour of the triangles changing to that of their new boundary.

Paint All assigns all the active triangles to the currently selected boundary.

Face painting is also controlled by an edge angle that is set at the bottom of the panel. This angle is a threshold value for the dihedral angle between the normals of adjacent triangles. Any edge with an angle greater than this is considered to be a "sharp edge." A face is then delimited not only by triangles of a different colour, but also by sharp edges. This makes the painting of the end face of a pipe, for example, much simpler. Selecting an appropriate edge angle, and clicking on a triangle on the end face can mark the face. The face is then identified by the program as bounded by the sharp edges, without needing to chop the model or mark lines.

Boundary Reduction

The 'Reduce' button will cause non-adjacent boundaries to be merged into one, bringing down the total number of boundaries. This operation will use all available boundaries, not just those that are currently active.

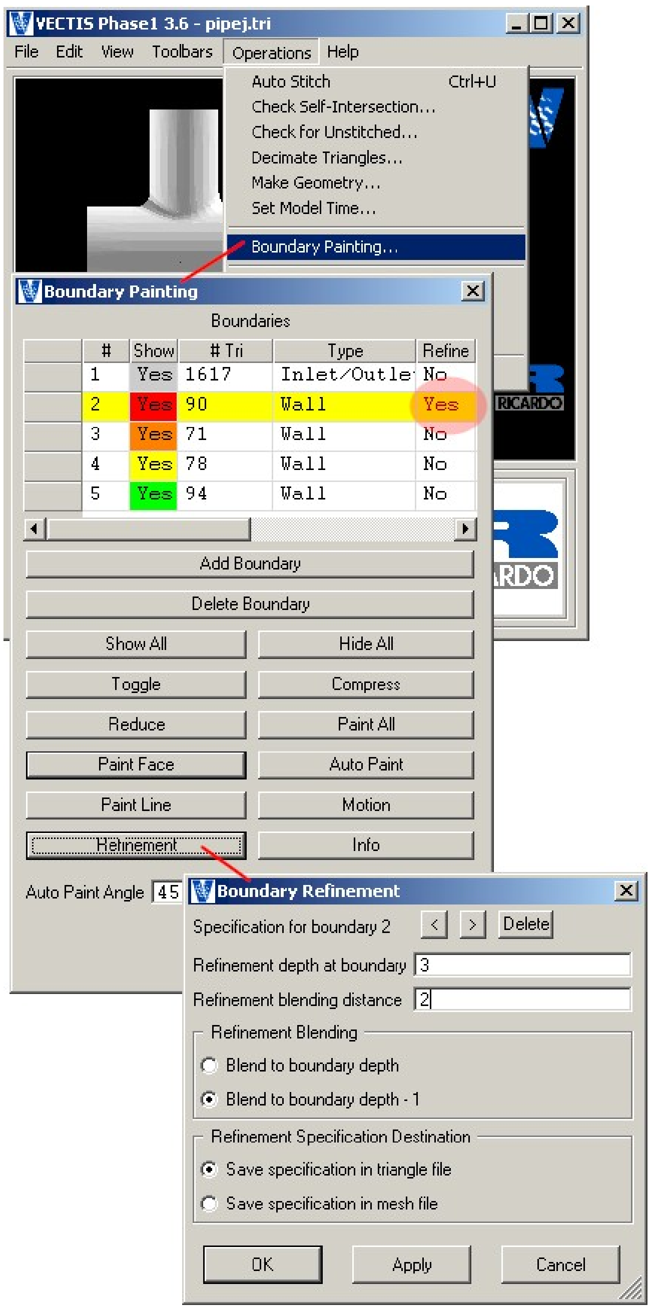
Auto Paint

This button paints boundaries automatically on the entire active part of the model, using the specified edge angle to define the limits of faces. The user is asked to confirm that the operation be continued or cancelled at this point. In a complex model, this may introduce a large number of boundaries. Using the 'Reduce' button can bring the number of boundaries back down.

3.15.3 Boundary defined refinement setup

VECTIS allows the user to control local mesh refinement based on boundary number.

The mesh refinement depth for cells adjacent to a given boundary can be specified by selecting the Refinement button in the Boundary Painting panel in phase1 as shown below.



Boundaries

#	Show	# Tri	Type	Refine
1	Yes	1617	Inlet/Outle	No
2	Yes	90	Wall	Yes
3	Yes	71	Wall	No
4	Yes	78	Wall	No
5	Yes	94	Wall	No

Auto Paint Angle 45

Specification for boundary 2

Refinement depth at boundary 3

Refinement blending distance 2

Refinement Blending
- Blend to boundary depth
- [✓ Blend to boundary depth - 1]

Refinement Specification Destination
- [✓ Save specification in triangle file]
- Save specification in mesh file

Opening the boundary refinement panel

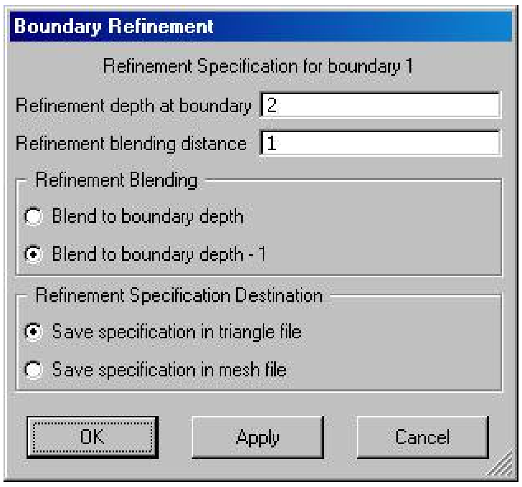
This brings up the following panel shown to the left.

The Refinement Depth specifies the refinement depth to be used in the global cells which are next to a particular universal boundary.

The Blending Distance specifies an integer which is used to control how the refinement at the boundary blends into the refinement level of the surrounding cells. The blend distance defines how many layers of cells there are to be at each forced refinement level when stepping back from the forced boundary refinement to the global mesh cell size.

Blending is achieved by giving the cells at the boundary a forced refinement level which is less than or equal to the specified refinement depth, and propagating away from the boundary in layers of successively lower forced refinement. The blend distance defines how many layers of cells there are to be at each forced refinement level.

Boundary refinement is applied after IJK refinement blocks (i.e. it will override IJK refinement blocks). However, the forced refinement level and refinement depth are only ever increased by boundary refinement specifications - never decreased.


Boundary Refinement

Refinement Specification for boundary 1

Refinement depth at boundary: 2

Refinement blending distance: 1

Refinement Blending

( ) Blend to boundary depth

(●) Blend to boundary depth - 1

Refinement Specification Destination

(●) Save specification in triangle file

( ) Save specification in mesh file

3.15.4 Saving the boundary refinement settings

The boundary refinement specifications can be saved in the triangle file, or in the mesh input file. This is controlled by the second radio button on the panel shown left. Where to save the specifications is a matter of user preference, and will depend on the particular application. If a boundary has a refinement specification in both the triangle file and the mesh file, the specification in the mesh file will take priority.

Boundary refinement specifications are written to the triangle or mesh input files by phase1.

For reference, the format of the information in these files is as follows:

□ Triangle file specification:

  SDF 2D integer array name:

  `VEC:BOUNDARY_REFINEMENT(3,NumberOfRefinementSpecs)`

  where the three elements per specification are:

  1 Boundary number, NB
  2 Refinement depth, DEEP
  3 Blending distance, BLEND (signed to indicate the "blend to" type, as above)

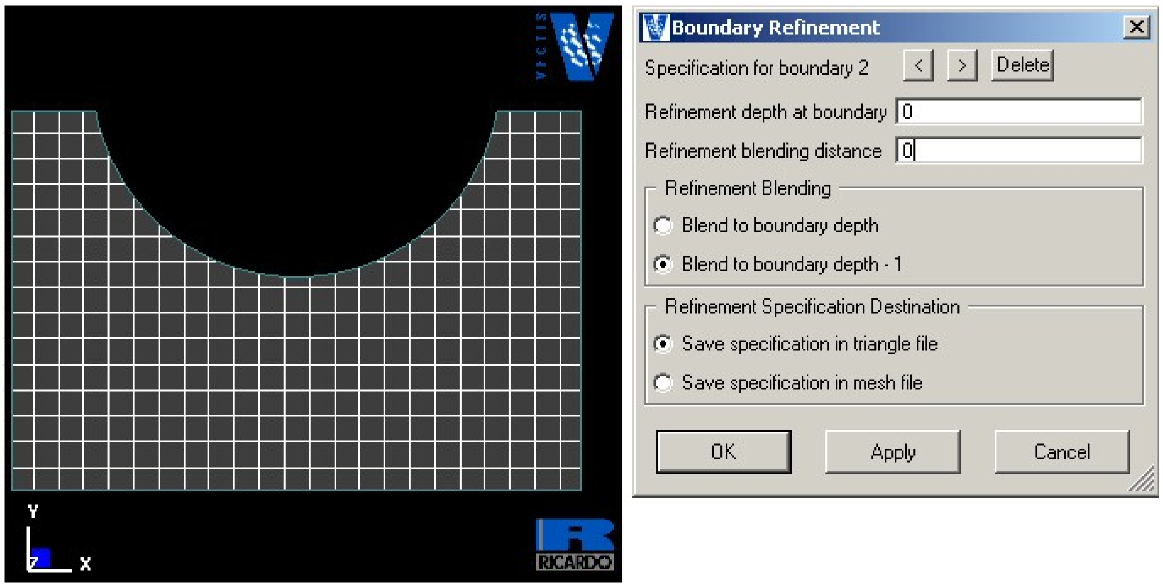
□ Mesh input file specification :- Any number of lines of the form :

  `BOUNDARY_REFINEMENT NB DEEP BLEND`

  where the variable meanings are as above.

3.15.5 Boundary Refinement examples

The effect of different parameters available for the surface refinement are shown below.



Boundary Refinement

Specification for boundary 2

Refinement depth at boundary: 0

Refinement blending distance: 0

Refinement Blending

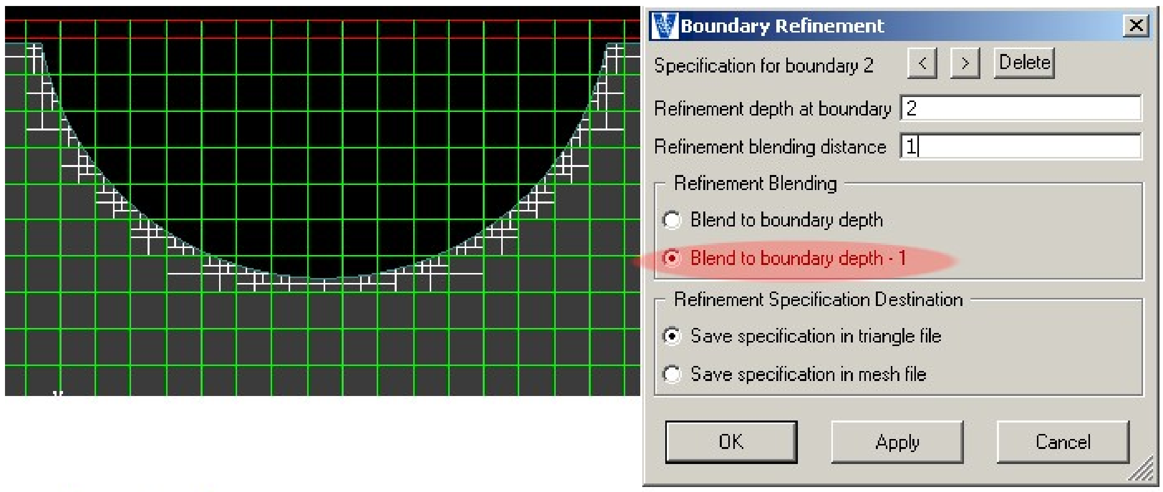
- Blend to boundary depth
- [✓ Blend to boundary depth - 1]

Refinement Specification Destination

- [✓ Save specification in triangle file]
- Save specification in mesh file

Surface refinement set to 0.

With this setup there is no surface refinement applied.


Boundary Refinement

Specification for boundary 2

Refinement depth at boundary: 2

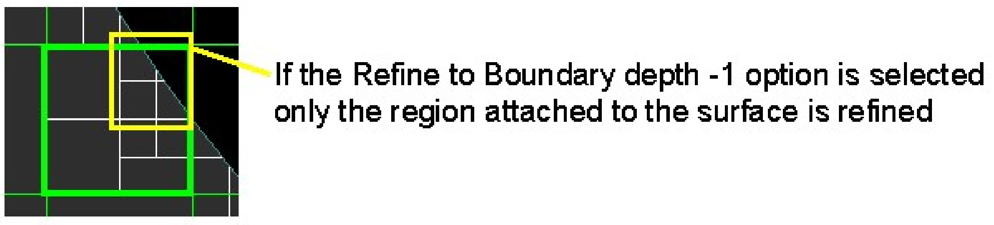
Refinement blending distance: 1

Refinement Blending

- Blend to boundary depth
- [✓ Blend to boundary depth - 1]

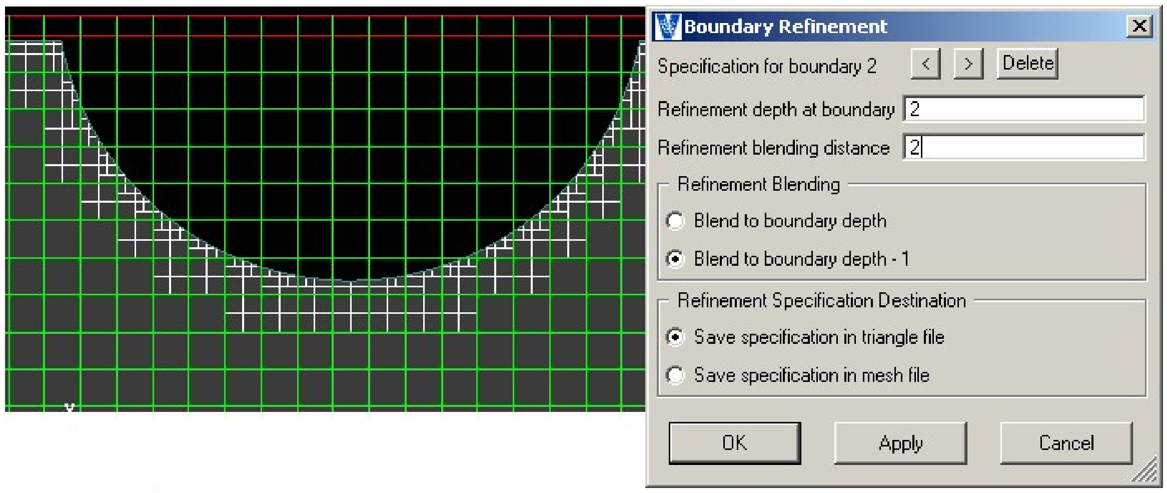
Refinement Specification Destination

- [✓ Save specification in triangle file]
- Save specification in mesh file


If the Refine to Boundary depth -1 option is selected only the region attached to the surface is refined

Surface refinement set to 2 with a blending distance of 1 and blend to boundary depth -1.

Now the surface refinement level is set to 2 and the cells next to the boundary are split into 4 smaller cells. Only the region of the global cell that is attached to the boundary are sub-divided since the blend to boundary depth -1 option is used.


Boundary Refinement

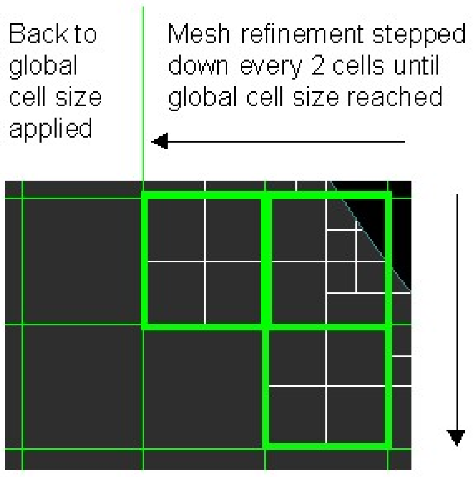
Specification for boundary 2

Refinement depth at boundary: 2

Refinement blending distance: 2

Refinement Blending
- ○ Blend to boundary depth
- ● Blend to boundary depth - 1

Refinement Specification Destination
- ● Save specification in triangle file
- ○ Save specification in mesh file

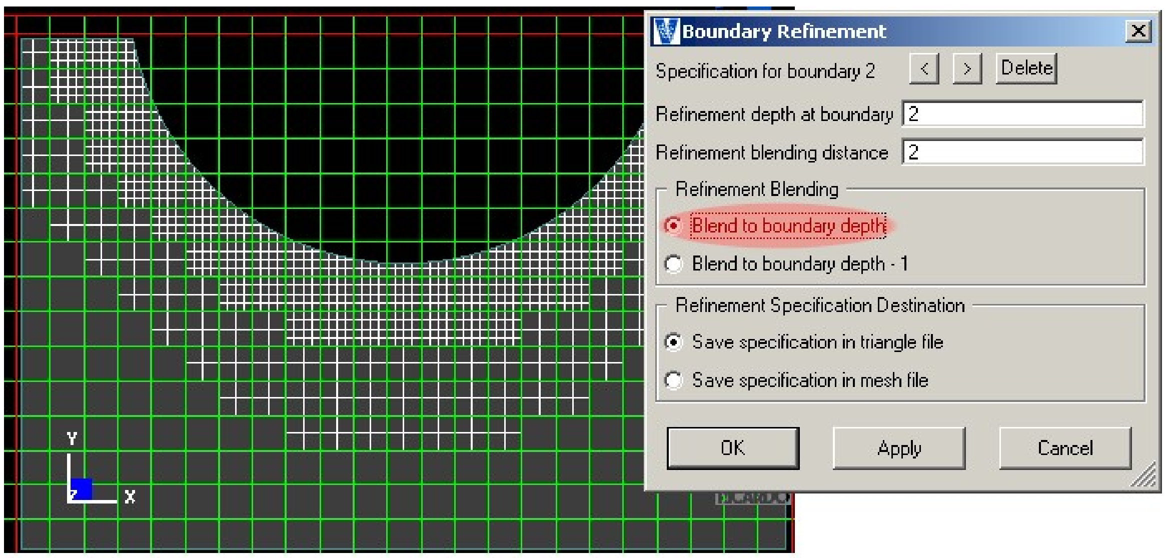


Back to global cell size applied
Mesh refinement stepped down every 2 cells until global cell size reached

If the Blend distance is set to 2 then the cell refinement is reduced by 1 every 2 global cells in each co-ordinate direction until the global mesh cell size is reached.

Surface refinement set to 2 with a blending distance of 2 and blend to boundary depth -1.

Now the Blend to boundary depth is set to 2 so the surface refinement is stepped back to the global mesh cell size using two global cells in each co-ordinate direction.



Boundary Refinement

Specification for boundary 2

Refinement depth at boundary: 2

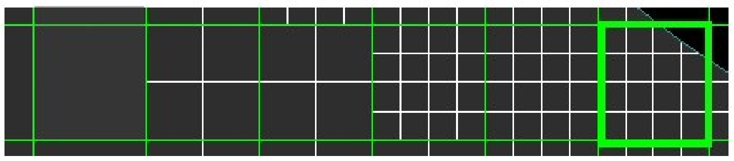
Refinement blending distance: 2

Refinement Blending

[✓ Blend to boundary depth] Blend to boundary depth - 1

Refinement Specification Destination

[✓ Save specification in triangle file] Save specification in mesh file



If the Refine to boundary depth is used the complete global is refined to the specified refinement level

Surface refinement set to 2 with a blending distance of 2 and blend to boundary depth.

Now the Blend to boundary depth option is used so that the complete global cells next to the boundary are sub-divided. The blend distance is still set to 2 so again 2 cells in each direction are used to step the mesh cell size back to the global cell size.

3.16 Mesh setup

A separate toolbar is provided with functions to set up the global mesh lines which the mesher uses as the basis for mesh generation. This toolbar also contains some viewing option commands that are likely to only be used when setting up mesh lines.

These are the top four buttons.

In order they are:

- X-Y Projection
- X-Z Projection
- Z-Y Projection

3D View

Only in the 3D view option may the model be manipulated, however mesh lines may only be edited in the first 3 2D projections. As the first button is automatically selected when the mesh set-up mode is entered, the model is put into the X-Y projection ready for mesh lines to be added.

The mesh is defined by a series of "main" mesh lines in three directions, with the spaces between these lines being divided evenly into any desired number of cells. The main lines are shown in red, and intermediate lines are shown in green. In the 3-D view, intermediate lines are not shown unless requested from the View Options panel.

The positioning of the main mesh lines and the number of divisions between main lines controls the cell density in different parts of the model. For any particular problem, the actual mesh density used will be governed by the requirement to resolve the flow accurately, and limited by the maximum problem size that can be analysed on the hardware available. In setting up the mesh, the user should be aware that VECTIS requires a "halo" of external cells around the active, or internal, cells in the calculation. To guarantee that this is achieved, there should always be a complete global mesh cell beyond the limits of the model in all six directions (Âśx, Âśy, and Âśz).

Mesh lines can only be defined while the viewing mode is one of the three orthogonal views. To add a new horizontal or vertical main line, click on the button with the appropriate single line.

insert a vertical mesh line.

insert a horizontal mesh line.

A line of the required type will become bound to the cursor and clipped at the limits of the existing mesh. Once the mesh line is approximately in the right position, clicking the left mouse button will place the mesh line. Multiple mesh lines may be placed in this manner.

To exactly place a mesh line, once it is on the canvas, click on the 'Place Mesh Line' button

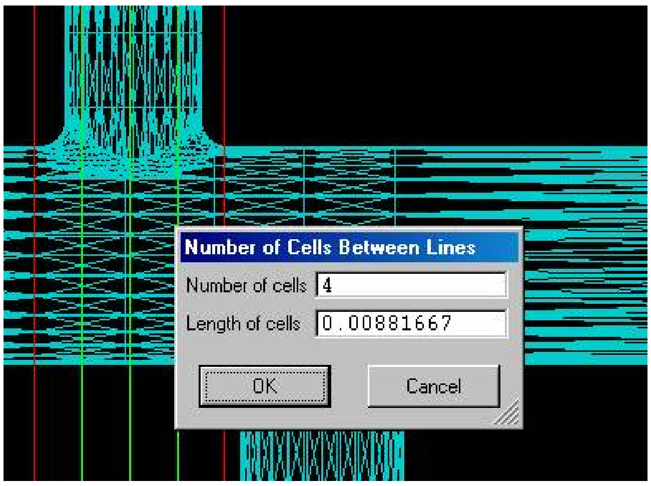
and type in the exact co-ordinate.

To specify the number of divisions between two main lines, the appropriate subdivision button should be selected:

create vertical sub-divisions.

create horizontal sub-divisions.

When the mouse cursor is moved over the model, a shaded area will appear between the main mesh lines that bound the cursor. When the left mouse button is clicked, a panel will appear to allow the number of subdivided cells that should appear in that shaded area as shown in the figure to the left.


Number of Cells Between Lines

Number of cells: 4

Length of cells: 0.00881667

Either the number of sub-divisions can be entered or a cell length. The cell length will be adjusted, when it is found to be approximate, to be the length that results in the nearest integer number of divisions.

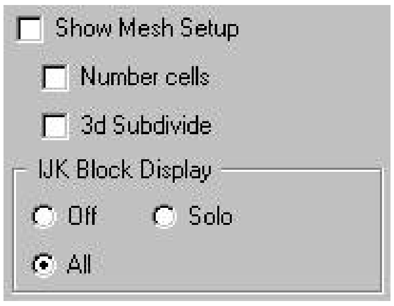
To remove a main mesh line, click on the 'Delete Mesh Line' button and click on the mesh line to be removed.

The main mesh line nearest to the point clicked will be deleted.

The 'Delete Mesh Line' button

deletes a main mesh line.

3.17 Mesh Set-up View Options

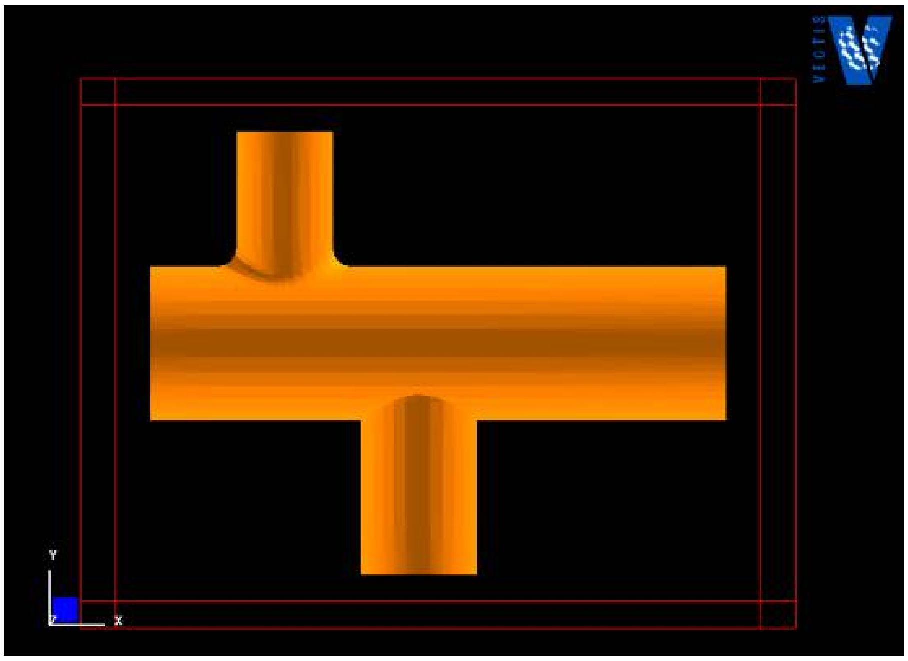

☐ Show Mesh Setup

☐ Number cells

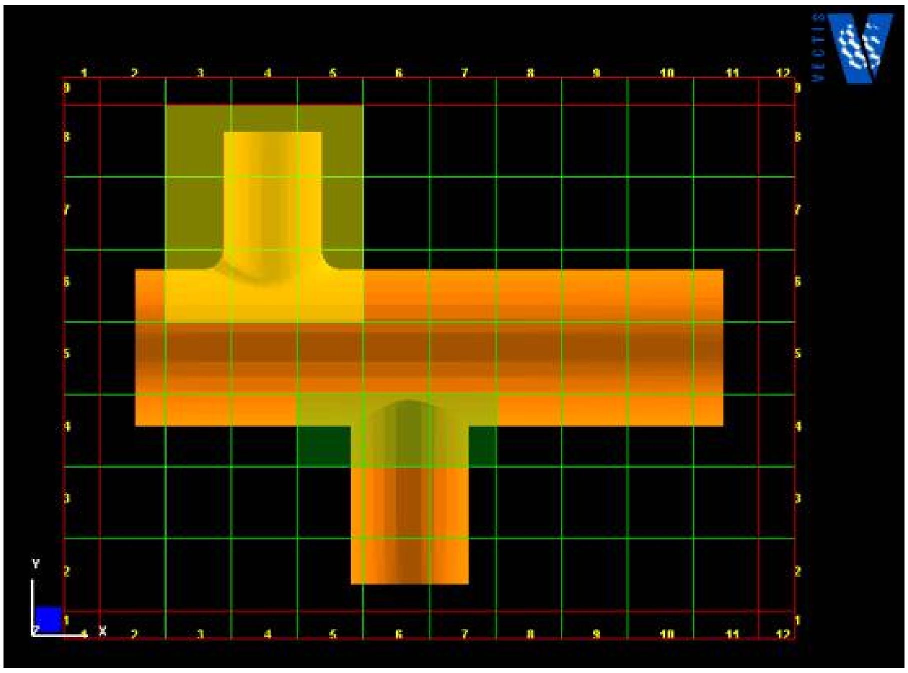
☐ 3d Subdivide

IJK Block Display: ○ Off ○ Solo ● All

Selecting the 'Show Mesh Set-up' option adds the outline of the mesh to the view of the model.



If a mesh has not been defined for the current model then a default mesh is created and displayed. This default mesh is the simplest mesh that satisfies the mesher requirement for a halo of cells around the model.



The 'Number cells' option will add the IJK indices of each of the mesh cells to the view.

The 'Show Mesh Set-up' option will allow the IJK blocks that may be defined during mesh set-up to appear in a 3d view. Normally, IJK blocks may only be seen in the 2D view projections to prevent the view from being cluttered.

The '3d Subdivide' option performs a similar function allowing the green mesh subdivision lines, that may also be defined in mesh set-up mode, to appear in a 3d view.

In 'Mesh Set-up Mode' the 'IJK Block Display' options can be used to switch off the IJK Block display (off), display the current IJK Block only (solo) or display all the IJK Blocks (all).

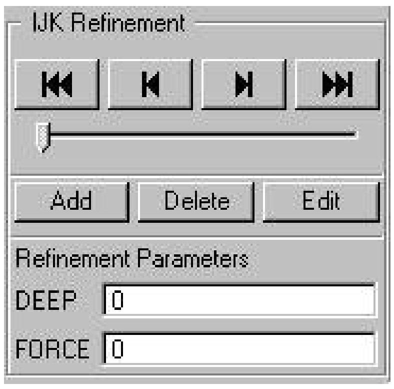
User Defined Mesh Set-up

The mesh set-up view options can also be set in the Vectis.cfg file which is read at program start.

3.17.1 IJK Refinement Blocks

A panel is provided, on the mesh set-up panel, for the setting of IJK blocks that control the level of mesh refinement in different parts of the mesh.

To use this panel, select the mesh tab or the mesh set-up button on the button bar or select the IJK Navigation option from the Operations menu.


IJK Refinement

Refinement Parameters

DEEP 0

FORCE 0

IJK Refinement block settings

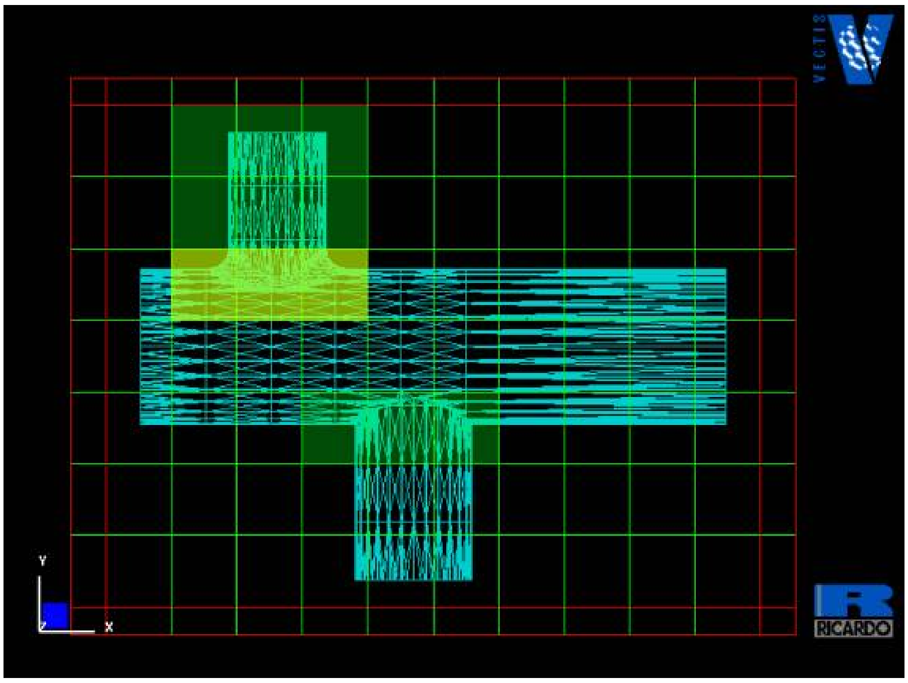
The Add button on this panel allows a new IJK block to be created. When the left mouse button is clicked on the canvas, a rectangle is drawn between the point chosen and the current cursor position. Clicking the left mouse button again defines the extent of the IJK block.

To completely define the IJK block, a different 2d view must be selected and the extent of the block in the third dimension defined. This may be done by clicking the Edit button, and dragging out the extent of the IJK block in exactly the same way as before.

The current IJK block is shown in yellow, and the current block may be cycled through the available blocks by using the VCR style controls or the slider on the panel. The Delete button will delete the current IJK block and the allowed and forced refinement depths for the current IJK block are set via the panel as shown in the Figure.

The refinement level of an IJK block is shown by the colour of the block on the canvas. A depth of 0 is shown in red, of 1 is shown in magenta, and a refinement depth of 2 (the default) is shown in cyan. Examples of all of these, and of the current block, are shown in the Figure.

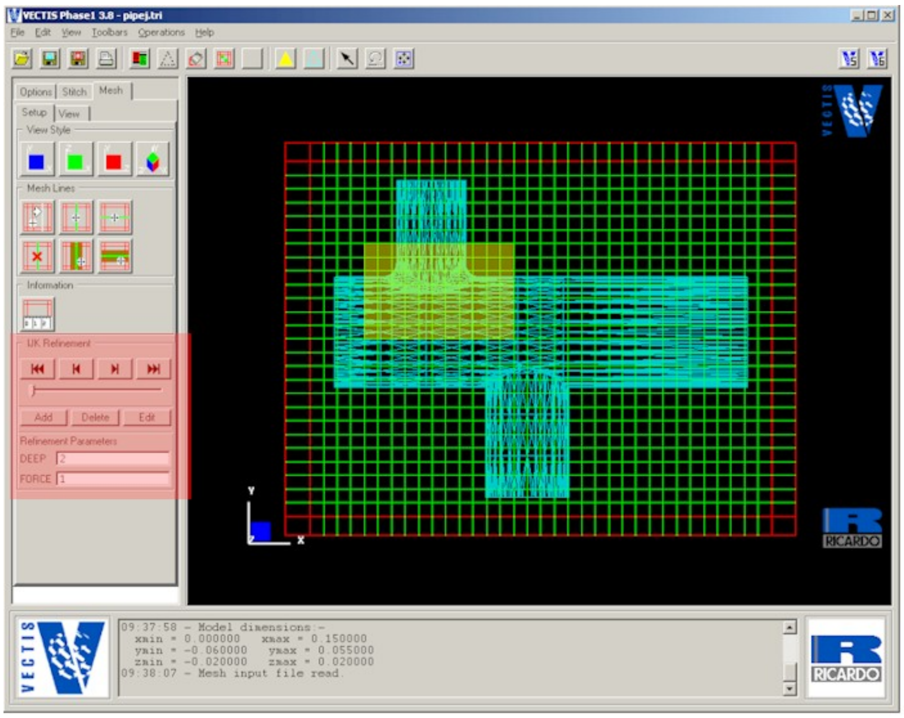
Note that the blocks are defined in XYZ co-ordinates, so that they do not move if the number of mesh lines is changed.



IJK Block Setup

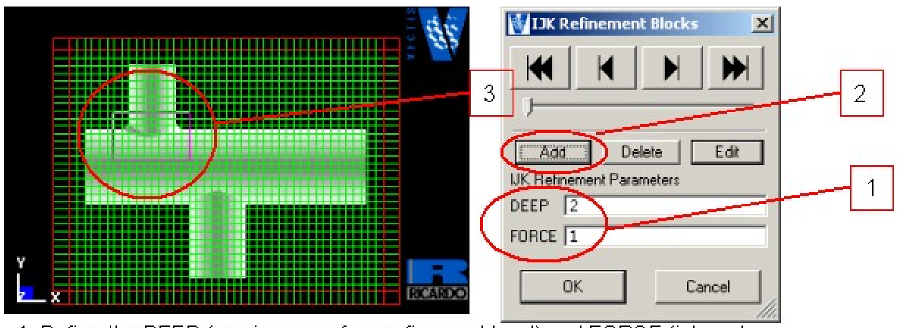
Example of how to define an IJK refinement block

Local refinement with in the mesh structure can also be set up using IJK refinement blocks. This are set up using the panel highlighted in red below .



```
09:37:58 - Model dimensions:-
  xmin = 0.000000   xmax = 0.150000
  ymin = -0.060000   ymax = 0.055000
  zmin = -0.020000   zmax = 0.020000
09:38:07 - Mesh input file read.
```

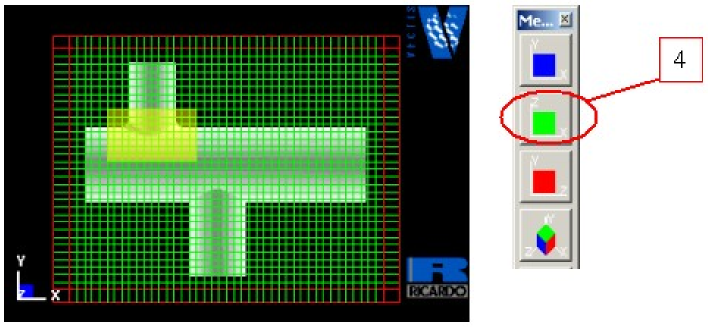
The process for setting up an IJK refinement block is shown below.



1. Define the DEEP (maximum surface refinement level) and FORCE (internal cell refinement level) values.

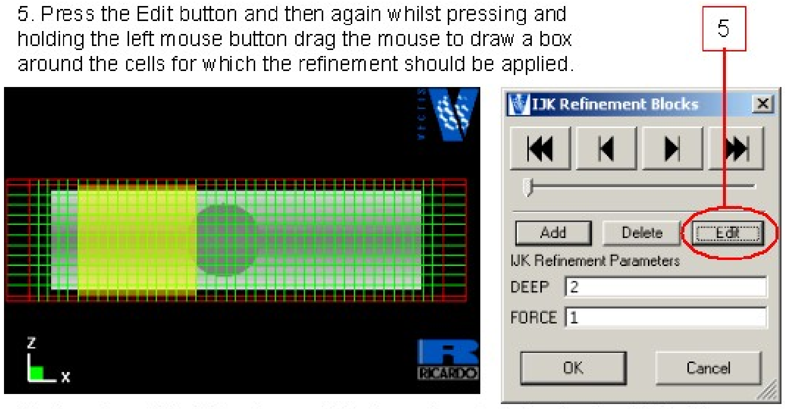
2. Press the Add button

3. Whilst pressing and holding the left mouse button drag the mouse to draw a box around the cells for which the refinement should be applied.



4. Change the view

5. Press the Edit button and then again whilst pressing and holding the left mouse button drag the mouse to draw a box around the cells for which the refinement should be applied.



The location of the IJK refinement block can be altered using the Edit button

The DEEP and FORCE cell side division values are applied in the same way as the global refinement.

IJK Cell Refinement DEEP and FORCE Options

Cell refinement is the capability for varying the refinement level across the mesh.

Each global cell can have associated with it:

1. Its maximum allowed refinement level, IDEEP
2. A "forced" refinement level, IFORCE

The forced refinement level specifies a level of refinement to be applied to the cell before any ordinary refinement due to the presence of the boundary takes place.

For example, if a cell has IFORCE=1 and IDEEP=2, it will first be split into a 2x2x2 set of cells, and refinement down another level will then be applied if the boundary passes through (or close to) the cell. Thus, it is possible to effectively have a finer mesh localised to a particular area (e.g. the valve bridge region in a coolant flow), which does not extend throughout the mesh in the x, y and z -directions.

Forced refinement should be used with care: level 1 produces 8 cells from each global cell; level 2 produces 64; level 3 produces 512.

Similarly, using too large a level IDEEP refinement should be avoided.

There can be an enormous increase in number of cells produced when using IDEEP=2, as compared to IDEEP=1. Depths greater than two should be used only with caution.

The way the refinement level is varied is done by specifying IDEEP and IFORCE for I/J/K blocks of cells.

This information appears in the MESH.INP file as follows:

```
IJK_BLOCK = is ie js je ks ke ideep iforce
```

and can be defined in the mesh setup section of Phase 1.

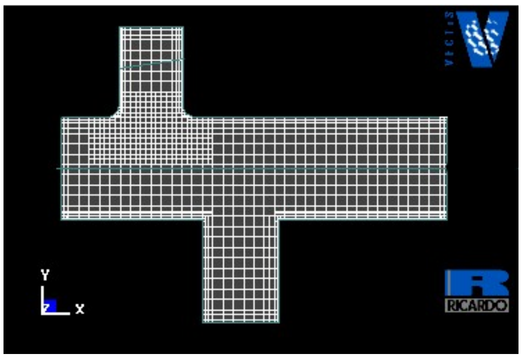
Any number of lines of this type can be put in the file, anywhere between the comment line (line 2) and the mesh coordinates.

Where defined blocks overlap, the values of IDEEP and IFORCE from later blocks overwrite the values from earlier ones.

$2^{DEEP} = 2^2 = 4$ for surface cells. Therefore the global surface cells can be divided into 4 cells in each co-ordinate direction.

$2^{FORCE} = 2^1 = 2$ for internal cells. All cells inside the IJK refinement block will be sub-divided by 2 in each co-ordinate direction.

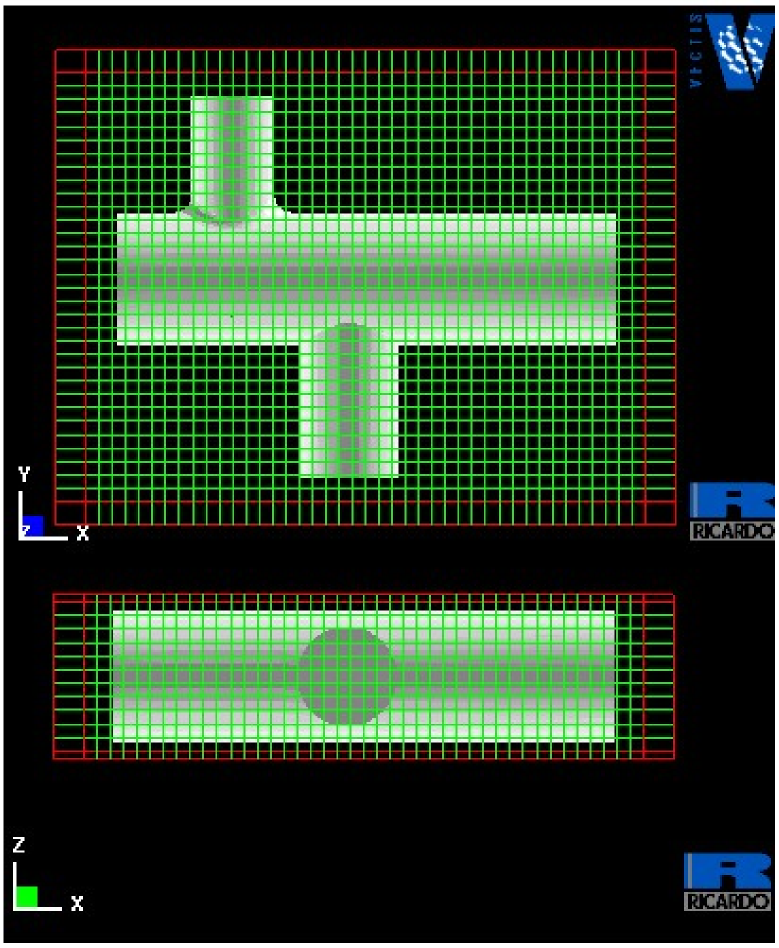
A slice of the computational mesh created using the control mesh shown above is shown below. The local refinement region can clearly be seen.



3.17.2 Control Mesh Setup Suggestions

The global mesh is used to define the required base cell sizes through out the model. An example control mesh is shown below.

An example control mesh defining the global cell size



The green lines are the global cell divisions and the red lines are fixed control lines. Note that two red lines must be completely outside the model extents in each direction.

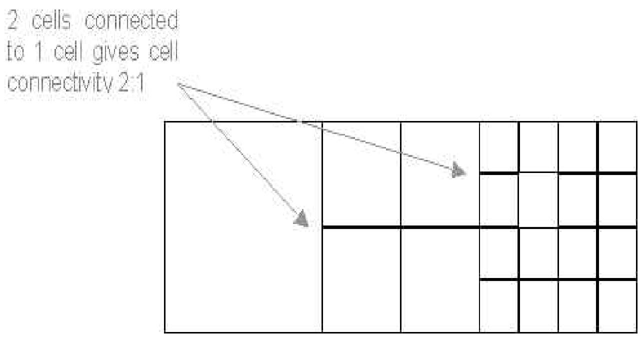
The reproduction of the surface detail is achieved by either ensuring that the global cells are small enough to capture the local geometry or by the use of surface cell refinement. Local IJK refinement regions can be setup in the detailed core flow locations to improve the accuracy of the CFD calculation.

The following suggestions are provided to improve the mesh setup.

Cell Connectivity

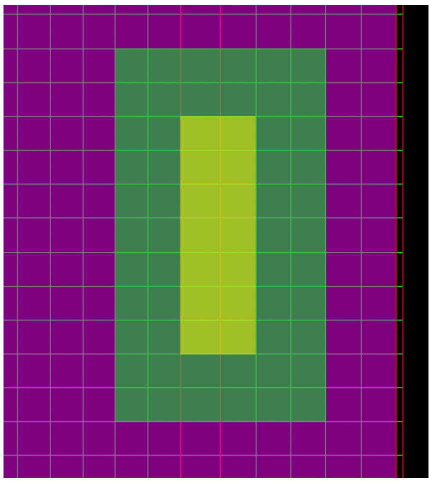
Cell connectivity is concerned with how the cells are linked to each other between refinement regions. The best practice is to ensure that the cell connectivity never exceeds 2 cells connected to

1 cell as shown by figure one below.



Cell connectivity.

Ensuring that the cell connectivity never exceeds 2:1 reduces the inaccuracy due to numerical error and artificial viscosity. This is best achieved when using the IJK refinement by having overlap regions of ideally at least two global cells. The figure below is a snapshot of a VECTIS control mesh that shows this setup.



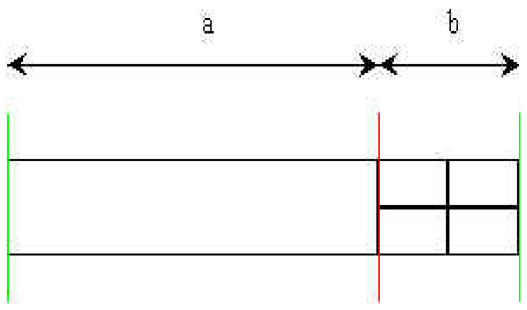
Refinement block overlap regions.

The different IJK blocks have different refinement settings so with the forced level of refinement increasing by 1 for each block from the outer block to the inner block

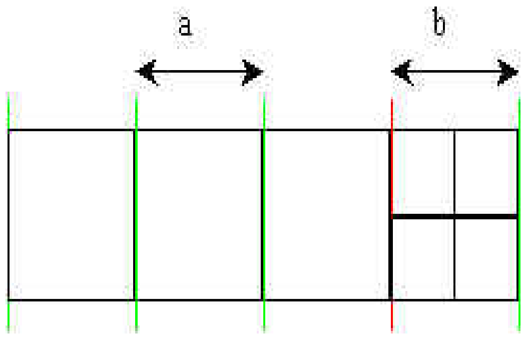
Cell shape

It is best to maintain square cells throughout the computational domain as much as possible since this improves the numerical accuracy.

This is particularly important with a mesh that includes IJK refinement regions. The figures below show the nonideal setup and the ideal setup.

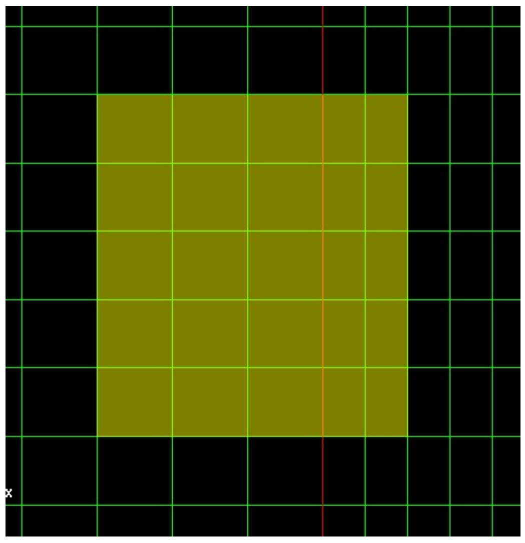


Neighbouring cells with significantly different aspect rations.



Neighbouring cells with suitable aspect ratios.

If it is necessary to have a variation in global cell size at the edge of an IJK refinement block then overlapping the refinement block will minimise the numerical error as much as possible. The figure below shows this concept.



Refinement region overlaps different global cell size regions reducing neighbouring cell size ratios.

3.18 Warning and Error Messages

Note that these pages are part of the ongoing improvements to the VECTIS documentation - please contact

`RS_Support@Ricardo.com`

for specific questions about certain warning or error messages.

In Phase 1 warnings and errors are identified with a four-digit number. These messages are written to the program information area.

The first digit identifies the type of message:

0	Warning
1	Array bounds error
2	Other error

Those messages which require explanation are listed and explained below. Others are self-explanatory.

WARNING 0102. VECTIS PHASE1.
THREE TRIANGLES FOUND ON LINE m - n
TRIANGLE NUMBERS i j k

This warning can be produced when loading triangle data into the triangle processing part of Phase 1. Three triangles have been found with an edge between the two nodes m and n. This is not permitted in the definition of a closed surface (which is that each edge should lie on exactly two triangles), so the situation is treated as two separate edges. Places where this occurs will appear as red lines (lines which need stitching). A large number of these warnings may indicate overlapping or duplicate surfaces.

WARNING 0103. VECTIS PHASE1
INCORRECT IDENTIFIER IN FILE filename
IT DOES NOT APPEAR TO BE A VECTIS TRIANGLE FILE

A valid triangle file must start with "!VECTIS_TRIANGLES". A file not containing this identifier is ignored.

WARNING 0201. VECTIS PHASE1.
POOR CONVERGENCE IN BISECTION SOLUTION
EXITING ROUTINE

This warning may occur during the tracking of VDA trim curves on trimmed surfaces. It is caused by round-off errors in the numerical scheme used to locate the intersection between a trim curve and a line of constant s or t on the surface. There are usually no adverse effects resulting from this, but a very large number of these warnings may indicate a poorly defined CONS curve in the VDA file.

WARNING 0202. VECTIS PHASE1.
NO TRIANGULATION FOUND FOR POLYGON ON FACE facename

The program has not been able to triangulate an individual polygon on a trimmed surface. This may result in holes in the triangulated surface. It may indicate a self-intersecting trim curve, or other poorly defined trim curve in the VDA file.

WARNING 0203. VECTIS PHASE1
NO VDA HEADER BLOCK IN FILE filename
EXITING ROUTINE

The file is not a valid VDA file, as it does not contain a correct VDA header block.

WARNING 0204. VECTIS PHASE1.
INCOMPLETE CONS LOOP CLOSED ON FACE facename

The VDA standard requires that a set of CONS curves form a closed loop in order to define a FACE (a trimmed surface). The program has detected a non-closed loop, but has determined that the loop can be closed by joining its end points. This warning indicates a file which does not conform to the VDA standard.

WARNINGS 0401 - 0413.

Various messages
These warnings indicate errors in the mesh coordinate section of the mesh input file. The action resulting from these errors is to produce a uniformly spaced mesh with the specified numbers of mesh lines in the i, j and k directions.

WARNING 0414. VECTIS PHASE1.
NUMBER OF MAIN "X" MESH LINES EXCEEDS MAXIMUM ALLOWED VALUE
LINE NOT CREATED.

The user has tried to insert a new mesh line in the mesh set-up section of Phase 1 which would cause an array overflow. Corresponding warnings 0415 and 0416 are for the other two directions.

ERROR 2201. VECTIS PHASE1.
CONS consname ON SURFACE surfname
DOES NOT START AT THE END OF THE PREVIOUS CONS
FACE WILL BE TRIANGULATED AS A SURFACE

and
ERROR 2202. VECTIS PHASE1.
CONS LOOP NUMBER n ON SURFACE surfname
HAS DIFFERENT START AND END POINTS.
FACE WILL BE TRIANGULATED AS A SURFACE

The VDA standard requires that a set of CONS curves form a closed loop in order to define a FACE (a trimmed surface). The program has detected a non-closed loop which can not be closed by joining its end points (see warning 0204). The trimming information is therefore discarded, and the untrimmed surface is triangulated.

ERROR 2206. VECTIS PHASE1.
ELEMENT elemname IS NOT A CONS ON SURFACE surfname
PROGRAM TERMINATING and
ERROR 2207. VECTIS PHASE1.
CONS consname FOR SURFACE surfname NOT FOUND
PROGRAM TERMINATING
and
ERROR 2208. VECTIS PHASE1.
ELEMENT elemname IS NOT A SURFACE.
PROGRAM TERMINATING.

These are errors in a VDA file associated with the naming and referencing of elements.

ERRORS 2501 - 2504

Various errors in the format of STL files.

3.19 The Vectis.cfg File

The VECTIS configuration file name vectis.cfg file can be placed in the working directory, the users home directory (UNIX/LINUX only) or the VECTIS config directory and is read at program start.

```
# Configuration file for Vectis
#
#path to logo files (png format)
#
LOGO_FOR_WHITE_BACKGROUND: (path to logo file)
LOGO_FOR_BLACK_BACKGROUND: (path to logo file)
#
# phase1 and phase6 view options
#
```

```
MODEL_CANVAS_LOW_REFRESH_RATE  ON    (ON/OFF)
#
# phase1 view options
#
PHASE1_FAST_RENDER: ON (ON/OFF)
PHASE1_SURFACES: ON (ON/OFF)
PHASE1_SURFACES_FLAT: OFF (ON/OFF)
PHASE1_SURFACES_FLIP: OFF (ON/OFF)
PHASE1_NUMBER_CELLS: ON (ON/OFF)
PHASE1_3D_SUBDIVIDE: ON (ON/OFF)
PHASE1_IJK_BLOCK: SOLO (OFF/SOLO/ALL)
PHASE1_OUTLINES: MULTI (OFF/MONO/MULTI)
PHASE1_HIGHLIGHT_PARTS: ON (ON/OFF)
PHASE1_HIGHLIGHT_HOLES: ON (ON/OFF)
PHASE1_HIGHLIGHT_SHARP_EDGES: ON (ON/OFF)
PHASE1_HIGHLIGHT_HOLE_NODES: OFF (ON/OFF)
PHASE1_MESH_VIEW_FACES: ON (ON/OFF)
PHASE1_MESH_VIEW: BOUNDARIES (BOUNDARIES/DOMAINS)}
```

4 MESHING

4.1 Introduction

VECTIS-MAX solver can read several formats of grids generated in different mesh generators. VECTIS-MAX offers its own mesh generator, called VMESH, which works automatically and produces a locally refined Cartesian mesh. The Figure 4.1 shows position of the mesher in VECTIS-MAX system and its communication with the other modules. Two files containing the input information need to be supplied to the mesher: trifile and meshfile. Both these files can be generated by the preprocessor (Phase1). Trifile (its usual extension is ".tri") is a SDF file containing a completely closed, fully connected triangulated surface. Meshfile (its usual extension is ".tri") is an ascii file containing the meshing control parameters. The format of this file will be described in section 4.3. The mesher produces final gridfile (its usual extension is ".GRD"), which contains the generated grid. In ".OUT" file, there is a copy of the messages printed to the screen. Also, VMESH temporarily creates several .aux files, which are removed at the end of the mesher's run. VMESH also contains tools which can help the user to automatically correct certain portion of problems in the input triangulated surface. These tools will be described in section 4.7.

4.2 How to Run VMESH

VMESH is a console application. To start it from the command line, type

```
vmesh meshfile [switches]
```

The format of the input ascii meshfile and possible switches are described in two following sections. VMESH produces gridfile whose name is derived from the name of the meshfile; .GRD extension is added (e. g. when port.mesh is used as the input file, the name of the output file will be port.GRD).

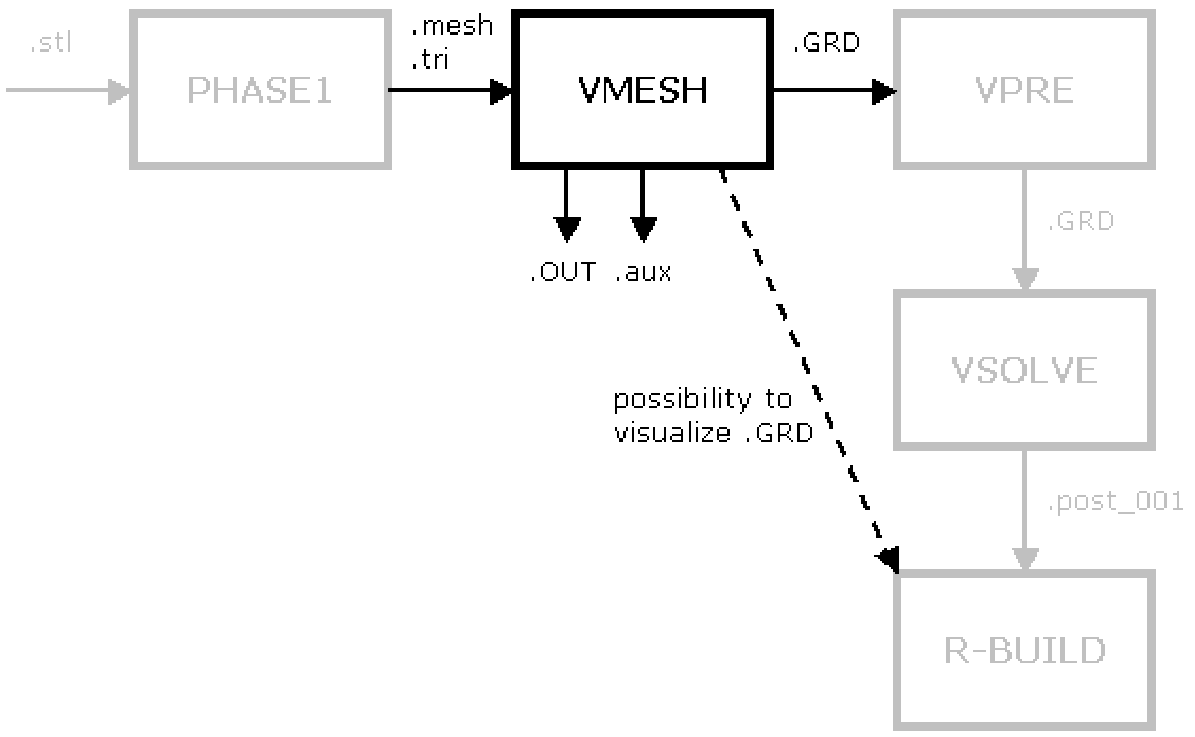


Figure 4.1: VMESH position is VECTIS-MAX system

4.3 Setting Up the Input File

VMESH is non-interactive and is controlled by the input file and command line options. The input file can be generated by Phase1 (see "MESH SETUP" section of Phase 1). Phase 1 writes the mesh line coordinates to the file, together with the default values for other parameters. The user needs to edit this file if values other than the defaults are required for any parameter.

The principal parameters for the user to check are the following:

MODEL_NAME The name of the SDF file (trifile) produced by Phase 1. This file contains the geometry of the model, the boundary information produced by the boundary identification section of Phase 1 and the boundary types information.

REFINEMENT_DEPTH The number of times a cell can be subdivided or refined (see section 4.6.2 below); default value is 2.

EDGE_THRESHOLD = angle The sharp edge criterion; it is a threshold parameter in degrees.

When angle between normal vectors of two adjacent triangles is greater then this threshold, the common edge will be considered as sharp feature. The default value is 35°. It is not recommended to set a value less than 20°, because then the number of generated boundary faces starts to increase rapidly.

IJK_BLOCK Definition of IJK refinement blocks (see section 4.6.2 below).

It is not recommended to try to manually modify the section MESH_COORDINATES, since its format is quite complex and it is easy to make a mistake here. The best approach is to let Phase1 to write the information down.

Since the meshfile generated by Phase1 must be readable also by the older versions of VECTIS (VECTIS 3), it may contain also three keywords which are not used by VMESH: PATCH_TYPE, VOLUME_DEPTH and OUTPUT.

4.4 Command Line Options

In this section, all switches which can be used with VMESH are described. When none of the parameters -test -rep -sep -info -locate is used, normal meshing work is assumed and the given filename is assumed to be the input ascii file.
vmesh meshfile [switches]

GENERAL SWITCHES:

-h or -help or - -help shows the help

-v shows the version of the executable

-elimill all ill cells detected during the meshing task will be removed

-dd switches on Distance Decimation method (which is used after each generation of patches by EF)

-nops switches off Polygon Simplification (all patches stay triangular)

-nocs disables Cell Splitting feature

-fmc b1 b2 ... bn forces Marching Cubes (see section 4.5) method for all boxes intersected with one of the listed boundaries b1 b2 ... bn. However, MC might be dangerous in the situations when the surface intersects an edge of the box more than once. Correct tying of patches is not assured in such cases.

-fef b1 b2 ... bn similar option to -fmc, but Exact Fit (see section 4.5) is forced instead of Marching Cubes

-info reads the given gridfile and prints its basic information

-o name the output will be written to files name.GRD and name.OUT instead of files with names derived from the input meshfile.

REFINEMENT RELATED SWITCHES:

-blendcontrol this switch can be used in cases when there are close surfaces with different blending and the user wants the blending to spread in one direction only (so as the surface B is not unnecessarily divided because of propagation of refinement from the surface A); usage of this switch can decrease the number of generated cell, but generation of boxes might be 10% slower

DELETING OF SMALL CELLS:

-smallio ratio sets the ratio SMALL_CELL_VOLUME/BOX_VOLUME (which defines the maximum volume of cell which should be considered as small) for all open flow boundaries (input and output) or cyclic boundaries; the default value is 1.0×10^{-3}

-smallint ratio sets the ratio SMALL_CELL_VOLUME/BOX_VOLUME (which defines the maximum volume of cell which should be considered as small) for all boundaries that form interface between domains; the default value is 1.0×10^{-6}

-smallb bouinx ratio sets the ratio SMALL_CELL_VOLUME/BOX_VOLUME (which defines the maximum volume of cell which should be considered as small) for all cells that contains patches from boundary bouinx. Option -smallio has higher priority in the case when it can also be used for the cell.

-small ratio sets the ratio SMALL_CELL_VOLUME/BOX_VOLUME (which defines the maximum volume of cell which should be considered as small) for all other cells (where -smallio or -smallb is not defined); the default value is 1.0×10^{-3}

SAFE POSITIONS OF MESHLINES:

-noduallevels switches off the technique of dual levels (when meshlines lie on different discreete levels than vertices of triangles; this technique ensures that there is no collision between sides of boxes and triangles); when dual levels are switched off, moving of meshlines to safe positions is automatically switched on

-mvfrac fraction sets fraction which defines the tolerance for moving of meshlines out from dangerous positions; the tolerance is calculated as $dist \times fraction$ (where $dist$ represents the distance of the tested meshline from the closer of the two neighbouring meshlines); this option can be used only with -noduallevels

POLYGON SIMPLIFICATION PARAMETERS:

-ps_sfang degangle sets the minimal angle from which the edge between two polygons is considered as sharp feature in polygon simplification. Default value is 35.0 degrees. When used, this value eclipses the value set in EDGE_THRESHOLD, which has the same meaning (see section 4.3).

-ps_ncang degangle sets allowed level of nonconvexness of output polygons generated by polygon simplification routine (inner angles of polygons can be 180°+ps_ncang). The default value for ps_ncang is 5.0°.

TEST GRID:

-test tests all cells in the generated grid; indexes of cells with the worst problems are reported. In order to use this command, the grid needs to be generated already. The supplied filename is expected to be the name of the gridfile

-verbosetest tests all cells in the generated grid; the indexes of all problematic cells are reported; it is also possible to combine this switch with -locate command (see below)

-verbosetest -locate meshfile passes the name of the input ascii meshfile, so as the -verbosetest could report not only indexes of the problematic cells, but also their IJK information.

BASIC VISUALIZATION TOOLS: (The proper tools for visualization of the mesh are implemented in R-Build. However, VMESH also contains some features which allow simple visualization of cells. Zoom, rotate and pan can be controlled in the same way as in Phase1. There are several hotkeys which can be used: *'l'* switches off/on visualization of lines of faces, *'k'* shows vertices which are scalable by pressing + and -, *'s'* switches on/off light, *'n'* toggles additional lines if drawn.)

-viewc CellInx reads VECTIS-MAX grid file, opens a GLUT window and visualizes the cell CellInx (in this case, the given filename is interpreted as the name of the VECTIS-MAX grid)

-viewcgrp Ncells Cell1 Cell2 ... reads VECTIS-MAX grid file, opens a GLUT window and visualizes Ncells number of cell CellInx (in this case, the given filename is interpreted as the name of the VECTIS-MAX grid)

-viewb BouToVis reads VECTIS-MAX grid file, opens a GLUT window and visualizes the boundary faces on boundary BouToVis (in this case, the given filename is interpreted as the name of the VECTIS-MAX grid)

-viewbgrp NBouToVis BouToVis1 BouToVis2 ... reads VECTIS-MAX grid file, opens a GLUT window and visualizes boundary faces on NBouToVis number of boundaries BouToVis1, BouToVis2,... (in this case, the given filename is interpreted as the name of the VECTIS-MAX grid)

-view reads VECTIS-MAX grid file, opens a GLUT window and visualizes all boundary faces (in this case, the given filename is interpreted as the name of the VECTIS-MAX grid)

-viewijk IS IE JS JE KS KE the geometry in the specified I,J,K area is visualized. For example, it is possible to run vmesh test.mesh -viewijk 2 2 54 54 25 25 to see the situation in the global cell 2 54 25. When 'n' is pressed, the global box is drawn as a wire model.

CHECK AND REPAIR INPUT GEOMETRY:

-rep (repair) this switch activates an iterative healing subroutine which attempts to find and correct degenerate parts of the geometry such as overlapped or intersected triangles. The resulting trifile is then stored with postfix _wt, both corrected and uncorrected triangles are painted as a boundary most relevant to their position. The mesher does not proceed to generate the mesh.

-sep (separate) this switch activates a thorough search of intersected and overlapped triangles; these are repainted as a new boundary and the whole geometry is then stored in a new trifile with postfix _wt. The user should check this trifile with Ricardo graphic tools (such as Phase1) and correct the problematic parts. The mesher does not proceed to generate the mesh.

-rep -sep When these two switches are used together, the mesher first attempts to heal problematic triangles, then repaints the remaining faulty triangles and stores the result in a new trifile with postfix _wt.

-not (no problematic triangle test) switches off the default rapid test of overlapped triangles; in this case, problematic triangles are not detected at all.

PREPARE GRIDFILES FOR MULTIDOMAIN:

-int b1 b2 ... bn

-interface b1 b2 ... bn This command informes the mesher that boundaries b1 b2 ... bn form the interface between domains. Patches on the interface will be treated differently (more simplified), so as the later work to make the grids conformal is easier. Also, the min-max extents of the model are not found on the geometry, but it is taken from the positions of mesh lines. This is necessary to ensure the same positions of meshlines when generating cells on the complementary meshes. Also, a different settings for deleting of small cells is applied here (see the option -smallint).

MAKE BOUNDARIES CONFORMAL (operates on finished gridfiles):

-conform GridfileA bA1 bA2 ... bAn GridfileB bB1 bB2 ... bBn When this switch is used, the two gridfiles GridfileA and GridfileB are loaded and all patches belonging to boundaries bA1, bA2, ..., bAn on the A geometry are tried to be made conformal to boundaries bB1, bB2, ..., bBn from the B geometry. The changed gridfiles are written down to files with postfix _conform. In order to perform this task successfully, it is necessary to use -int (-interface) command when generating both gridfiles before using this tool.

-conformrew GridfileA bA1 bA2 ... bAn GridfileB bB1 bB2 ... bBn When this option is used, the same actions are performed as in the case of -conform. The only difference is that the given gridfiles GridfileA and GridfileB will be rewritten (instead of creating *_conform.GRD files)

-cb_tolrnlevs number_of_levels This option controls the tolerance used for linking corresponding vertices of the two complementary geometries. The tolerance is calculated as the diagonal from the bounding box of the two geometries divided by number_of_levels. The default value of number_of_levels is 1.0×10^7.

-cb_tolr tolerance If this option is used, the tolerance used for linking corresponding vertices is directly set and not calculated from the diagonal length of the bounding box (see description of the option -cb_tolrnlevs).

-viewnonconform [minx maxx miny maxy minz maxz]

-vnc [minx maxx miny maxy minz maxz] When one of these synonyms is used together with -conform or -conformrew, the situation is visualised only (instead of trying to make the complementar boundaries conformal). The conformal patches are drawn as blue polygons, nonconformal parts are grey. Patches that are recognized to be alone (they have no possible corresponding patch to be tied on) are drawn as red polygons. If red polygons appear, the input specification should be checked. If the six values representing limits of the area are added, only part of the geometry will be shown. In order to learn hot keys for manipulation with the visualization, see "basic visualization tools" above.

TEST TRIFILES FOR MULTIDOMAIN MESHING (operates on two trifiles):

-cmpb TrifileA bA1 bA2 ... bAn TrifileB bB1 bB2 ... bBn This feature can serve for detection of problems in definition of interfaces for multidomain meshing. The common interface must be exactly the same in both input trifiles and this tool can check whether it really is. When this switch is used, the two trifiles TrifileA and TrifileB are loaded and for all vertices/triangles belonging to boundaries bA1, bA2, ..., bAn on the A geometry are searched corresponding vertices/triangles on the boundaries bB1, bB2, ..., bBn from the B geometry. The summary of the comparison is printed.

-cmpbw TrifileA bA1 bA2 ... bAn TrifileB bB1 bB2 ... bBn This switch is similar to -cmpb, but in the case that there are some topological or geometrical differences, two trifiles TrifileA_tested and TrifileB_tested are generated. The triangles are separated into several temporary boundaries - corresp_tri, problem_of_topology and problem_of_geometry. If there is a possibility to repair the differences, than also the trifile TrifileB_corrected is generated. The corrected triangles are marked as a boundary named corrected_tri.

4.5 Basic Scheme of VMESH

When normal meshing task is run (none of the parameters -test -rep -sep -info -locate is used), the scheme of the work looks like this:

1) READ INPUT FILE FOR MESHING TASK The input ascii file (meshfile) is read.

2) READ SURFACE TRIANGLES The input SDF file (trifile) is read. For each triangle, the geometrical extents in x, y, z are found. Then, the node to triangles connectivity is gathered. Also, the connectivity between the triangles is found. Then, normal vectors of triangles are prepared and area of each triangle is calculated.

3) PREPARE DUAL LEVELS In order to avoid collisions of sides of boxes with triangles perpendicular to x, y or z axes, meshlines are shifted to some discreet levels. The whole used space is divided to 4.2 billions of discrete levels on which new nodes can be created. For meshlines, a bit rougher grid is used: they can be shifted to each 8-th level. Also vertices of input triangles are shifted to each 8-th discreet level; however, those levels lie exactly in the middle of the levels used for the meshlines.

4) RAPID DETECTION OF OVERLAPPED TRIANGLES Defaultly, a rapid algorithm for detection of overlapped triangles is run here. If there are some problematic triangles, warning #1800 is printed together with the list of indexes of the triangles. If some problematic triangles are detected, the mesher will not stop. Usually, VMESH automatically overcomes small problems in the input triangulated surfaces. However, if it happens that the final gridfile contains cells with low quality (-test option detects problems), the user should try to improve the flaws in the surface and run VMESH again. See more information in the section 4.7.

5) PREPARE SHOEBOXES Shoeboxing is a system heavily used in several Ricardo products that helps to quickly limit number of elements that need to be taken into account when intersection tests are performed. The 3D space is divided to NI x NJ x NK boxes (so called shoeboxes). Then, for each input triangle all shoeboxes that are intersected by it are found. The index of the triangle is remembered by all found shoeboxes. This information will help later when intersections of a segment with surface triangles need to be found, because only triangles linked to the shoeboxes intersected by the segment need to be taken into account.

6) CONSISTENTLY ORIENT TRIANGLES Since the orientation of the input triangulated surfaces is random in the trifile, it is necessary to orient them so as the normal vectors of the triangles always point inside (into the flow domain).

7) PREPARE IN/OUT STATUSES Here, in/out statuses of the vertices of the global boxes (which are defined by meshlines) are found by ray method. This method inheres in counting intersections of the segment connecting a point with known in/out status and the tested point, with surface triangles. If the number of intersections is odd, the in/out status of the tested point will be opposite; when the number is even, the in/out status stays the same.

8) GENERATION OF BOXES There is a special section below (4.6) which describes in detail how the boxes are generated.

9) PREPARE VELOCITY LOCATIONS The common rectangular sides of the generated boxes are called velocity locations. According to which direction they are perpendicular, U, V and W velocity locations are distinguished (perpendicular to x, y and z, respectively). The velocity locations serve for navigation through boxes and for generation of inner faces on them. In this part of the algorithm, the generated boxes are indexed first. Then, velocity locations are generated.

10) GENERATE CELLS ON BOUNDARY BOXES For all boxes, it is decided whether the box is intersected by the surface, fully inner or outer. The optimal order of processing boundary boxes is found (smaller boxes need to be processed first to be always sure that when processing a box that has more than one neighbour, the neighbours are all done already). Than, all boxes intersected by the boundary are looped and their faces are generated in these steps:

POLYGON GENERATION PART

A1) Generate boundary faces (patches) If a sharp edge is detected in the box or if there is more than one boundary conditions or if one of the edges of the box is intersected more than once, Exact Fit method of generation of patches is used. It means that the triangles intersecting the box are broken to get the triangles exactly cutting the box. In the other cases (there is just a simple cut), Marching Cubes method is used (there are 14 basic patterns how to create triangles on the intersections of edges).

A2) Polygon Simplification This technique simplifies the patch structure (storing boundary faces as polygons is more memory efficient than keeping triangles).

A3) Generate inner faces if neighbours are already processed Loops all velocity locations of the box and on those locations where patches of the two adjacent boxes were already generated from both sides, the patches are tied (not to have gaps between boundary faces) and polygons of inner faces are generated. If any generated inner face is concave, it is split to convex parts here.

CELL ASSEMBLING PART

B1) Find a complete box Find whether there is a box that has all its polygons already generated. If there is no such a box, continue with a new box (go to A1).

B2) Distinguish separated volumes Find connectivity of faces and paint the polygons to find separated volumes in the box.

B3) Cell splitting On each separated volume, test whether there are concave features. If there are, split the cell.

B4) Delete small cell Test cell volume and if it is too small, remove it (inner faces need to become boundary faces of neighbouring cells).

B5) Save faces The generated boundary and inner faces are saved to auxiliary files (.aux) to be retrieved back later when assembling the final grid. Continue with B1.

11) FINISH GENERATION OF COMMON FACES The velocity locations of those cells which are fully inner need to be processed to generate their inner faces.

12) PRINT STATISTICAL DATA OF GENERATED MESH Now, the statistics of the generated grid is printed. Here is an example of such a report:

```
Number of generated cells: 9816 (boundary: 7168 ; internal: 2648)
--- Patching method ---
Number of cells processed by Marching Cubes: 6076 (84.53 %)
Number of cells processed by Exact Fit: 608 (8.46 %)
Number of volumes broken by Cell Splitting: 20 (0.28 % of boundary cells)
```

```
     [4 attempts to split a volume failed (16.67 % of all attempts)
      4 attempts gave incorrect number of volumes
      0 attempts gave volumes with too low quality, so the undo was applied]
  --- Cell quality ---
  Number of correct boundary cells: 6704 (93.27 %)
  NO PROBLEMS with negative volumes
  NO PROBLEMS with gaps
  There were 484 small volumes, they are deleted now (6.73 %)
  Number of cells which had to be deactivated: 484 (6.73 %)
  Total mesh volume: 1.49073e-09 u3
  [ = 6.40121e-10 u3 (2648 inner cells) + 8.50607e-10 u3 (6704 boundary cells)]
  -----------------------------------------------------------------
```

If there were problems with generation of a cell, warning appears here. Problems like this are nearly always linked to topological problems on the input surface. See section 4.7 which summarizes how to detect the problematic IJK spot.

13) **WRITING THE MESH FILE** Finally, the auxiliary files (.aux) are read, the output arrays are assembled and the output gridfile (.GRD) is written down. Then, the .aux files are deleted.

4.6 Generation of Boxes

This section describes how the mesh generator creates boxes (in which future cells are constructed). It includes an explanation for the user to understand the different options available in the input file. The process is illustrated in Fig. 4.2 to Fig. 4.5 in 2-D, though it actually operates in 3-D.

4.6.1 Box Generation Procedure

The starting point is the surface definition and the set of mesh line positions defined in Phase 1 (see Figure 4.2). VMESH calculates which global boxes have any volume inside the model (see Figure

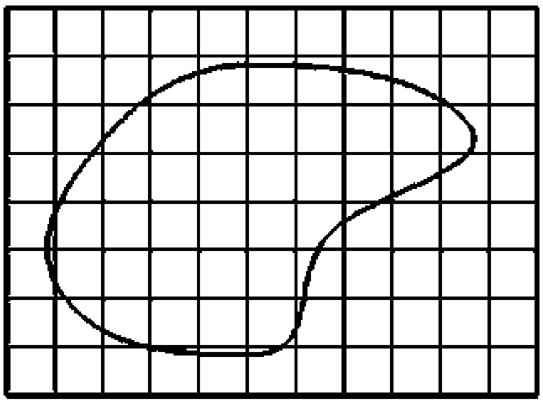


Figure 4.2: Start point of the mesh generation

4.3). As shown in Figure 4.4, the program then subdivides (refines) boxes intersected by the input

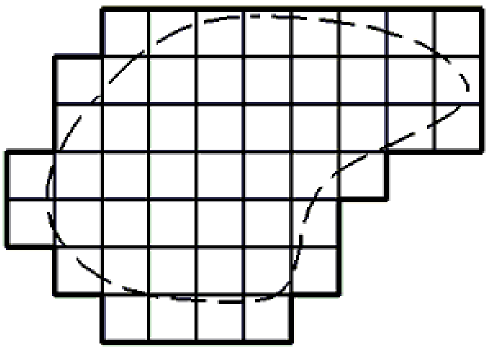


Figure 4.3: Global boxes containing volume

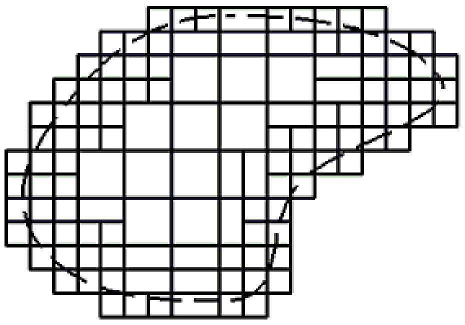


Figure 4.4: Refinement of global boxes

surface, to produce an accurate fit to the original shape. Figure 4.5 visualizes the final generated grid. The boundary boxes are cut by the surface and polygons of boundary faces are generated (here in 2D, the boundary faces appear to be segments).

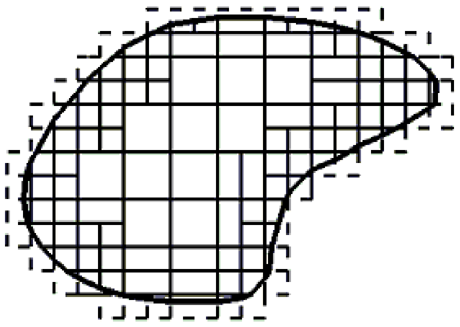
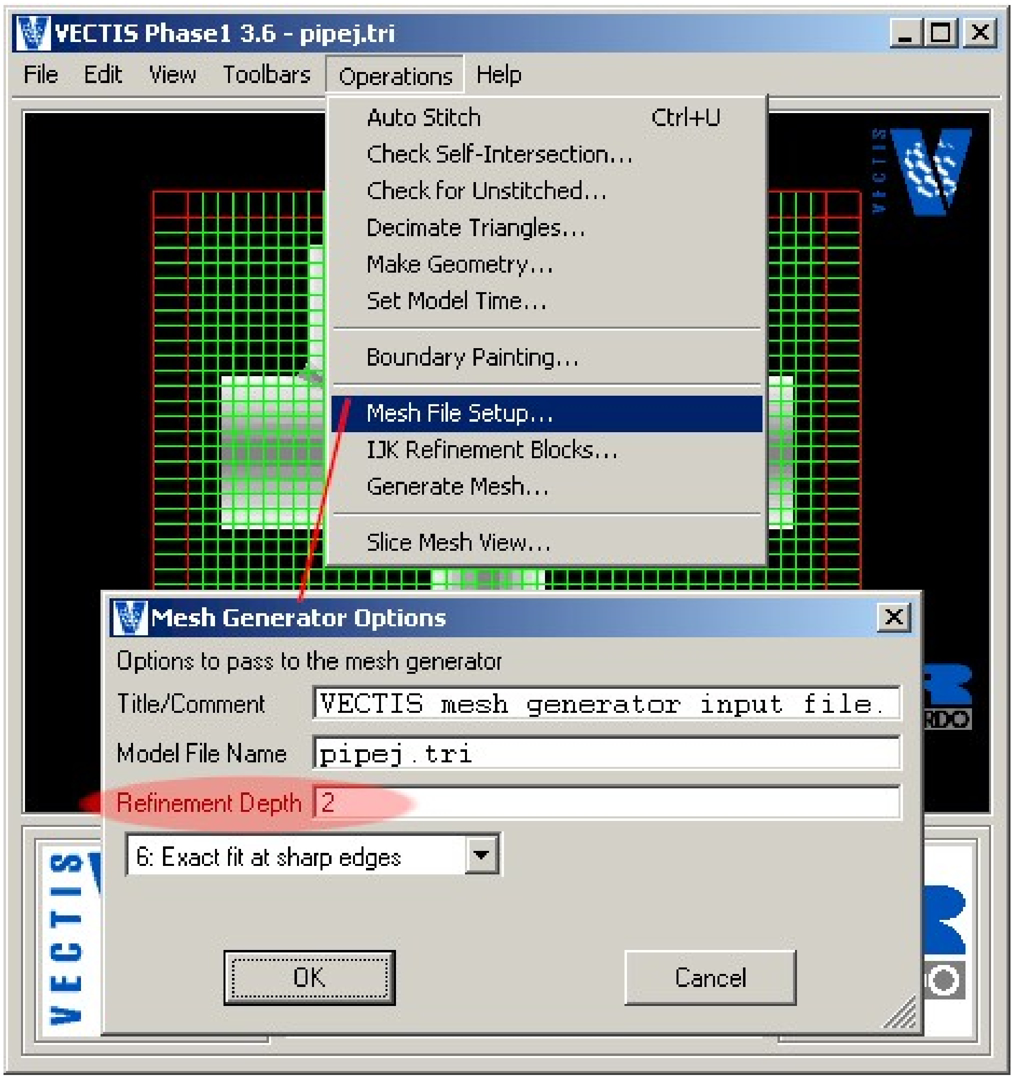


Figure 4.5: Final grid (all polygons are generated)

4.6.2 Parameters controlling generation of boxes

There are three ways how to affect refinement of global boxes:

1) Global refinement depth refers to how many times a cell can be split into two. This option can be set in the input ascii meshfile (see description of REFINEMENT_DEPTH in section 4.3). Figure 4.6 shows the panel in Phase1 which can be used for setting the global refinement depth. Refinement depth (RD) defines the maximum possible division of global box to subcells, which is $2^{RD}x2^{RD}x2^{RD}$. Therefore, a refinement depth of 1 allows boxes to be split into at most 2x2x2 refined cells, and a depth of 2 allows boxes to be divided into at most 4x4x4 refined cells. A depth of 0 does not permit any refinement. Boxes with different refinement levels are illustrated in Figure 4.7.



Options to pass to the mesh generator

Title/Comment: VECTIS mesh generator input file.

Model File Name: pipej.tri

Refinement Depth: 2

6: Exact fit at sharp edges

Figure 4.6: Definition of global refinement depth in Phase1

2) IJK refinement block allows to set different refinement level to a rectangular block of global boxes. The block can be defined within Phase1; the information is then stored to input meshfile.

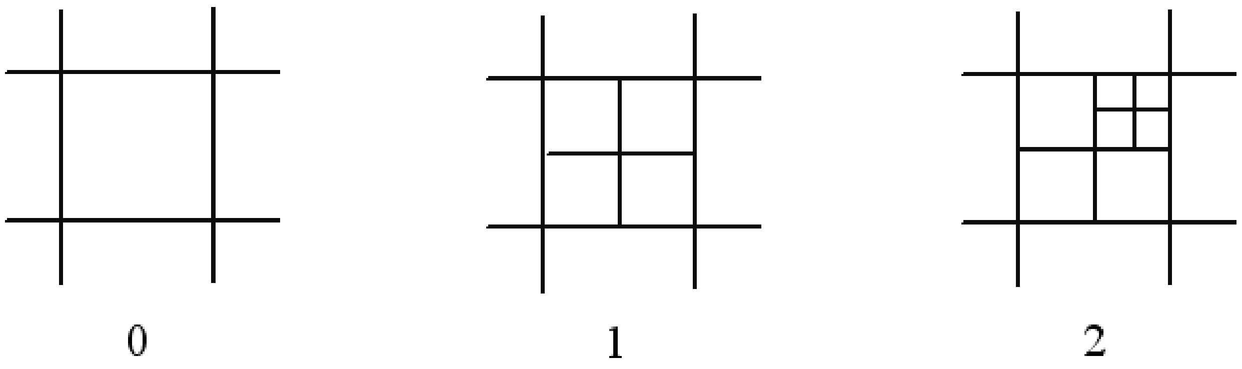


Figure 4.7: Global box divisions with different depths of refinement

The format is following:

```
IJK_BLOCK = is ie js je ks ke ideep iforce
```

For each this block defined as IS, IE, JS, JE, KS, KE (where S stands for start and E stands for end), values IDEEP and IFORCE are defined.

IDEEP is maximum allowed refinement level

IFORCE is a "forced" refinement level (the global box is going to be split without testing whether the input surface intersects it or not)

For example, if a box has IFORCE=1 and IDEEP=2, it will first be split into a 2x2x2 set of boxes, and refinement down another level will then be applied only if the input surface intersects the box. Thus, it is possible to effectively have a finer mesh localised to a particular area (e.g. the valve bridge region in a coolant flow), which does not extend throughout the mesh in the x, y and z directions.
The forced refinement should be used with care: level 1 produces 8 boxes from each global box; level 2 produces 64; level 3 produces 512. Similarly, using too large a level of IDEEP refinement should be avoided.

3) Boundary refinement specifies the refinement depth which is to be used in the global boxes which are intersected by a particular boundary. With this type of refinement, three variables can be set for a boundary:

Refinement depth at boundary refinement level prescribed to the boundary

Refinement blending distance specifies an integer value which is used to control how the refinement at the boundary blends into the refinement level of the surrounding boxes. Blending is achieved by giving the boxes at the boundary a forced refinement level which is less than or equal to the specified refinement depth, and propagating away from the boundary in layers of successively lower forced refinement. The blend distance defines how many layers of boxes there are to be at each forced refinement level.

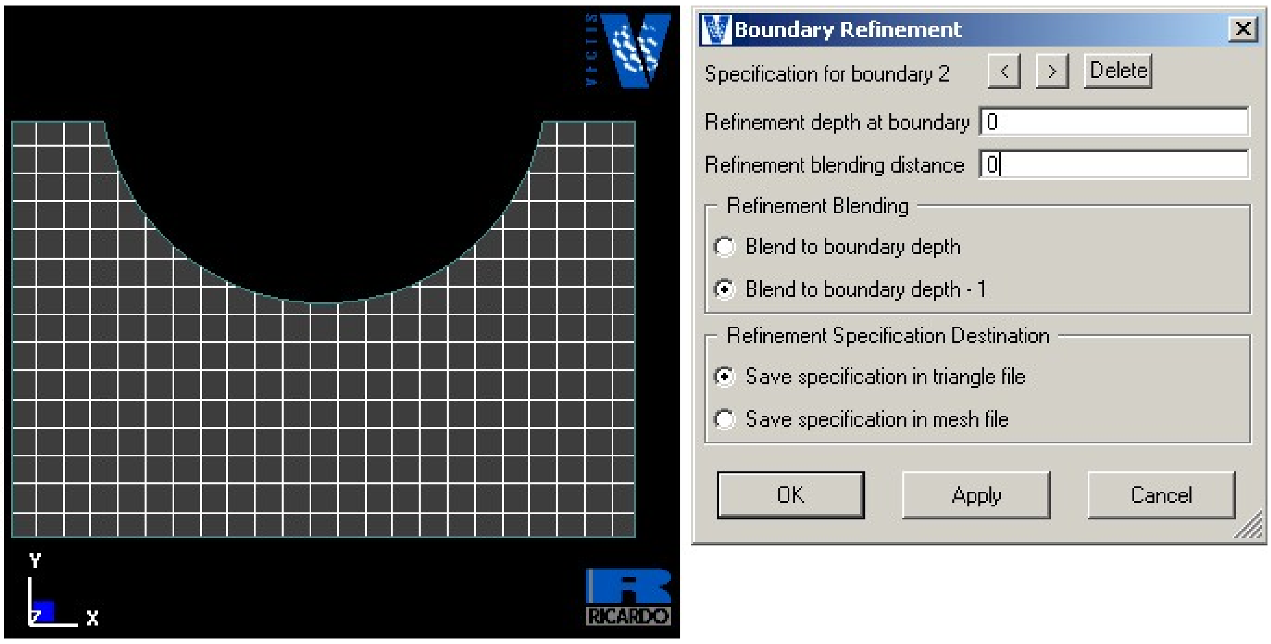
Blend to boundary depth -1 a boolean information (yes/no), which specifies whether the blending should start from "refinement depth -1".

Boundary refinement is applied after IJK refinement blocks (i.e. it will override IJK refinement blocks). However, the forced refinement level and refinement depth are always only increased by boundary refinement specifications, never decreased.

In Phase1, boundary refinement specification can be stored to the trifile or meshfile (the input ascii file). In trifile, the information is stored as 2D integer array VEC:BOUNDARY_REFINEMENT as three values for each boundary: BN (boundary number), DEEP (refinement depth) and BLEND (blending distance; signed as negative value where "Blend to boundary depth -1" is switched on. In meshfile, the specification is

```
BOUNDARY_REFINEMENT NB DEEP BLEND
```

There are several examples of combinations of boundary refinement depth and blending distances on figures 4.8 - 4.11. Comparison of figures 4.10 and 4.11 can shed light on meaning of the option "refinement depth -1": both cases have refinement depth 2, blending distance 2 and the only difference is "refinement depth -1" on/off.



Specification for boundary 2

Refinement depth at boundary 0

Refinement blending distance 0

Refinement Blending

Blend to boundary depth [✓ Blend to boundary depth - 1]

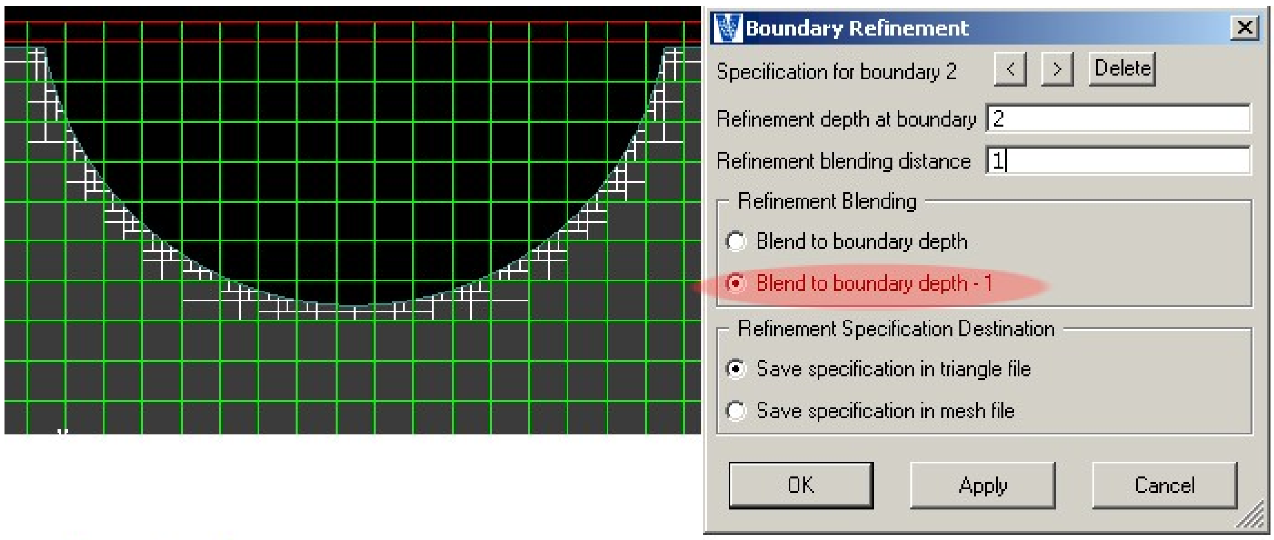
Refinement Specification Destination

[✓ Save specification in triangle file] Save specification in mesh file

Figure 4.8: Boundary refinement depth 0, blending distance 0

4.7 Problems With Quality of Input Surface

In comparison with VECTIS 3, the solver of VECTIS-MAX is more powerful. However, its demands for quality of used grids are higher. Therefore, in VMESH program, a lot of effort is spent


Boundary Refinement

Specification for boundary 2

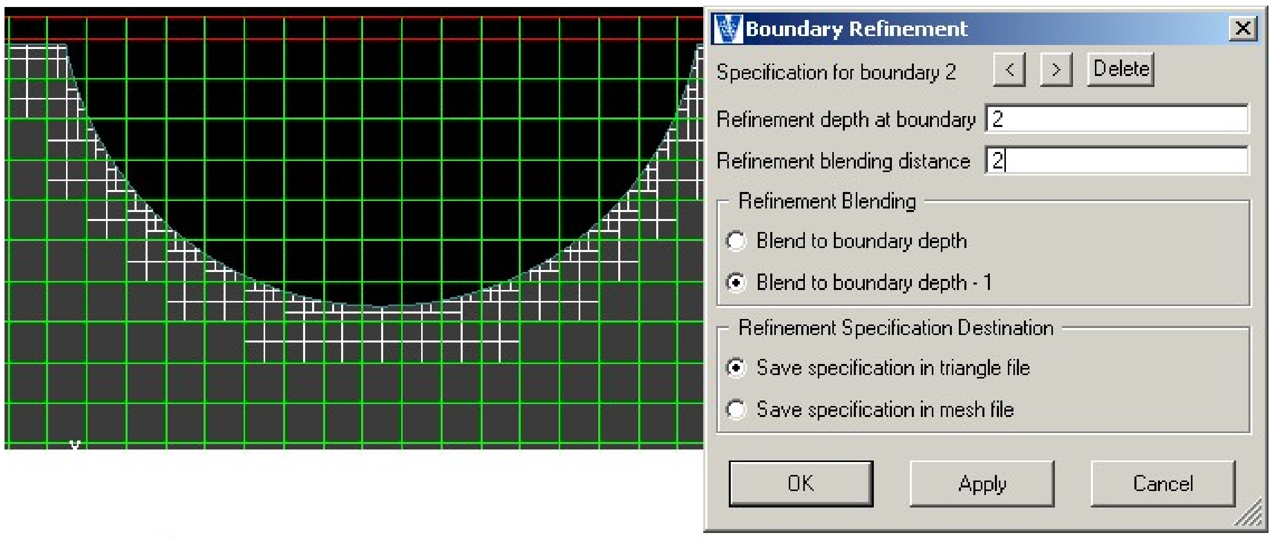
Refinement depth at boundary: 2

Refinement blending distance: 1

Refinement Blending
- Blend to boundary depth
- [✓ Blend to boundary depth - 1]

Refinement Specification Destination
- [✓ Save specification in triangle file]
- Save specification in mesh file

Figure 4.9: Boundary refinement depth 2, blending distance 1, "RD-1" option set on


Boundary Refinement

Specification for boundary 2

Refinement depth at boundary: 2

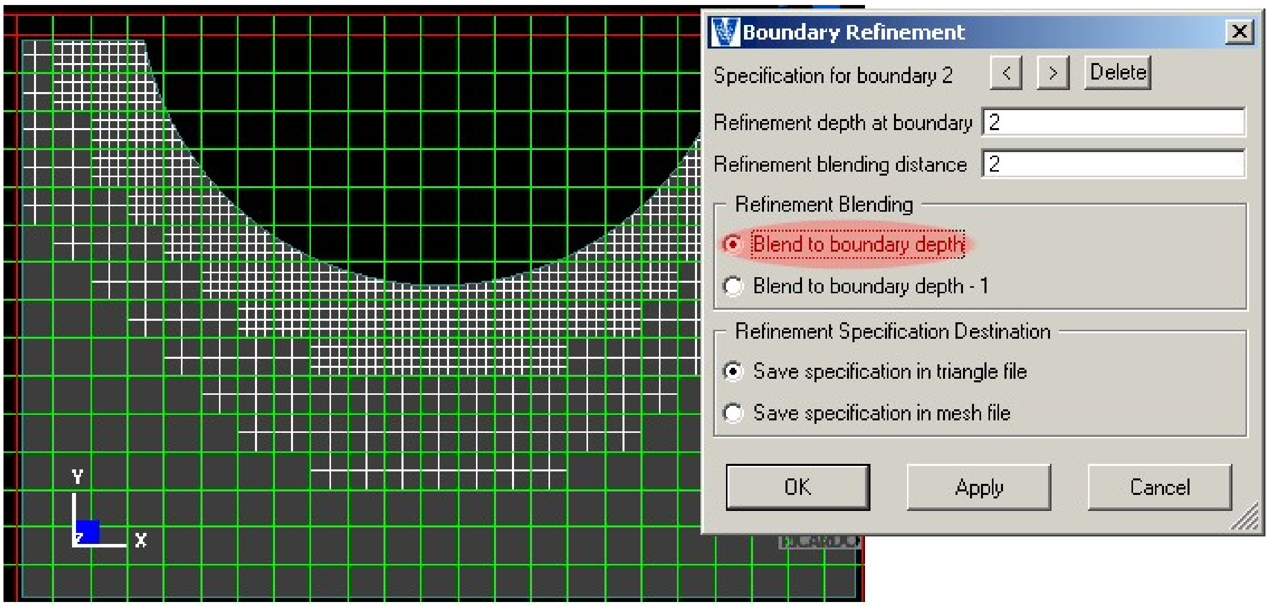
Refinement blending distance: 2

Refinement Blending
- Blend to boundary depth
- [✓ Blend to boundary depth - 1]

Refinement Specification Destination
- [✓ Save specification in triangle file]
- Save specification in mesh file

Figure 4.10: Boundary refinement depth 2, blending distance 2, "RD-1" option set on

to ensure that the generated cells fulfil certain quality criteria. VMESH has no problems to generate cells automatically, when the input surface is correctly defined. However, in real geometries imported from CAD systems there usually are flaws. For the mesher, it is essential that the user resolves serious problems such as gaps and intersections of triangles. However, even when there are no unstitched lines and test of intersected triangles reports no problems, small errors might remain in the surface. In Phase1, it is possible to recognize these problems according green edges of triangles. VMESH works quite robustly, so it can overcome majority of problems with the surface imperfection. However, it might happen that due to problems on the surface, the generated grid has some problematic cells. In this section, the possibilities of dealing with problems of the input triangles are summarized.
A reasonable procedure of mesh generation might look like this:



Boundary Refinement

Specification for boundary 2

Refinement depth at boundary: 2

Refinement blending distance: 2

Refinement Blending
- [✓ Blend to boundary depth]
- Blend to boundary depth - 1

Refinement Specification Destination
- [✓ Save specification in triangle file]
- Save specification in mesh file

Figure 4.11: Boundary refinement depth 2, blending distance 2

1) Remove gaps and intersections This work should be done in the preprocessor (Phase1). However, VMESH offers a possibility to automatically detect and repair overlapped and intersected triangles. The detection is activated when **-sep** option is used (see section 4.4). The problematic parts will then be painted as a new boundary. If the user wishes to automatically repair as many problems as possible, (s)he can use the command line option **-rep**. The algorithm detects all problematic triangles, removes them and tries to fill the gaps automatically. When the gap cannot be easily capped, the triangles are locally restored; such a problem needs to be resolved manually later. Since these automatic reparations might be dangerous in some cases (they can change geometric features), VMESH also offers a possibility to visualize the automatically repaired parts of the geometry. Usage of **-rep -sep** causes painting of all new triangles as a new boundary, so as the user can visually check correctness of the performed changes.

2) Run the mesher When the geometry is sufficiently clean, the mesher can be started. VMESH performs a simple check of overlapped triangles (can be switched off by **-not**). If there are any, their indicies are printed and the mesher continues. Usually, even though there are some overlapped triangles detected, the mesher has no problems to correctly generate the grid.

3) Check whether the generated grid is correct If there is a serious flaw in the input geometry, the mesher can stop its run with error
ERROR #1507 The closed loop cannot be gathered for box Box (ii=I, jj=J, kk=K, bx==box)
In the case of smaller problems, the grid is correctly generated, but when the test **-test** is performed (run *vmesh geometry.GRD -test*), problematic cells are reported.

4) If problems, find IJK, repair geometry and run again If the generated gridfile cannot be used, it is necessary to find the I J and K of the global box and check the part of the geometry inside it. In the case when the grid was not generated because of ERROR 1507, the IJK is already reported. If the problem was detected by running **-test** option, the global boxes can be found by using **-verbosetest -locate** option. For example when the mesher is run as

```
vmesh port.GRD -verbosetest -locate port.mesh
```

indicies of all problematic cells in port.GRD grid are gathered and corresponding IJK boxes are found according the information about meshlines stored in the passed meshfile port.mesh. The positions are found according geometrical position of vertices of the problematic cells. Then, user should check triangles in all reported global boxes.

4.8 Meshing for Multidomain Simulations

In comparison with normal meshing task for one domain only, preparation of conformal meshes for a multidomain simulation requires some additional effort. All domains that take part in a multidomain calculation are meshed separately and than a tool that ensures conformal faces on interfaces needs to be applied. The basic requirements for this meshing are these:

1) **Meshlines of all parts must be exactly the same** All the complementar meshes need to use exactly the same set of meshfiles. In order to check whether this condition is fulfilled, -cmpb command line option can be used. If the problems are small, the mesher can try to repair it when asked by -cmpbw option. For more details see description of these two command line options in 4.4.

2) **The used refinement must be exactly the same** It is essential to have the same refinement set in both geometries for the global boxes intersected by interfaces. The same refinement depth should be set for both geometries; also boundary refinement depth on interfaces must be the same. When IJK refinement blocks or boundary refinement depth with blending distance are used in a geometry, it must be set carefully so as the boxes created after refinement are exactly the same on the interfaces.

3) **Quality of input triangles** The normal meshing process can usually cope with small flaws in the input geometry (imperfections like folded triangles or triangles of nearly zero area). However, for multidomain simulations, it is recommended to pay higher attention to the quality of triangles defining the interface. The better quality of interfaces, the higher probability that the tool making boundaries conformal can complete its task successfully.

4) **Exact triangles defining interfaces** Interfaces on all complementar meshes need to be defined by exactly the same triangles. Also, it is not possible to have the situation that triangle defined as interface triangle does not have any corresponding triangle on the complementar mesh. Also, all the corresponding triangles must be marked as interface (no non-interface - interface couples).

5) **Mesher needs to be informed about interfaces** When preparing meshes for multidomain, the mesher needs to be aware which boundaries are interfaces (-int or -interface command line option needs to be applied), because these boundaries are treated differently. Patches on the interface are more simplified, so as the later work to make the grids conformal is easier. Also, the min-max extents of the model are not found on the geometry, but it is taken from the positions of mesh lines. This is necessary to ensure the same extents of boxes when generating cells on

complementary meshes. Also, a different settings for deleting of small cells is applied here (see the option -smallint).

6) Post-meshing tool ensuring conformal faces When all meshes for multidomain calculation are generated, it is necessary to use the tool of the mesher that ensures conformal faces on interfaces between each couple of meshes (-conform or -conformrew command line options). This tool compares boundary faces of common parts of the two given meshes and performs some actions to ensure their conformality (actions like movement of node, removal of node, split of edge, removal of edge, removal of face or split of face).

An example of meshing of a multidomain case consisting of two parts internal and external can look like this:

```
vmesh internal.mesh -int 12 13
vmesh external.mesh -int 12 13
vmesh -conform internal.GRD 12 13 external.GRD 12 13
```

When these commands are completed, new gridfiles internal_conform.GRD and external_conform.GRD should be created. The boundaries forming the interface of the two grids should be conformal.

4.9 Warnings and Errors

There is a list of warnings and errors which can occur during the run of VMESH. The majority of them can appear only under very special circumstances. Great deal of them are caused by problems in the input geometry. If it happens that a problem occurs but the geometry in the reported global box is without flaws, it might point to a problem in the algorithm. In such a case, supplying the problematic geometry to Ricardo Support Centre is recommended.

ERROR 1000 Fatal problem with IN/OUT test *(There is a problem with in/out test. The input geometry is probably not clean enough.)*

ERROR 1001 IN/OUT test inconsistency detected *(There is a problem with in/out test. The input geometry is probably not clean enough.)*

ERROR 1002 Fatal problem with IN/OUT test *(There is a problem with in/out test. The input geometry is probably not clean enough.)*

ERROR 1003 Fatal problem with IN/OUT test *(There is a problem with in/out test. The input geometry is probably not clean enough.)*

ERROR 1004 A flaw in triangle topology detected; check triangle *n (Unexpected topological situation near the triangle n was detected. Check the geometry around the triangle.)*

ERROR 1005 The triangulated surface cannot be harmonized. There must be a flaw causing Mobius strip effect near the triangle *n (Unexpected topological situation near the triangle n was detected. Check the geometry around the triangle.)*

ERROR 1200 Exact fit routine: not enough memory

ERROR 1201 Marching Cubes routine: zero or even number of intersections *(Marching Cubes routine encountered the situation, when there is zero or even number of intersections between two points with different in/out statuses. From the logic of the matter, the number of intersections here must be always even. There is probably a serious error in the input geometry.)*

ERROR 1202 The file *filename* cannot be opened

ERROR 1203 Problem with syntax found when processing the keyword *keyword (The keyword is known, however, its additional parameters cannot be properly read. The syntax of the keyword needs to be checked.)*

ERROR 1204 Unknown keyword: *keyword*

ERROR 1205 Routine FindNdInOut: problem with evaluation of in/out status of a velocity location *Location (There is a problem with in/out test on the velocity location Location. The input geometry is probably not clean enough.)*

ERROR 1206 Generation of faces: incorrect in/out test detected on location *location (There is a problem with in/out test. The input geometry is probably not clean enough.)*

ERROR 1207 Cannot read the grid file *filename*

ERROR 1211 Distance Decimation routine: not enough memory

ERROR 1212 Polygon Simplification routine: not enough memory

ERROR 1213 Incorrect version of meshfile detected *(There is an incorrect version written in the meshfile. The first version valid in VECTIS-MAX is !VECTIS_MESH_INPUT_V1.100 .)*

ERROR 1500 Problem with generation of face on velocity location *location (On the given velocity location, there is a problem with generation of internal face. This problem might be caused by unclean geometry near the velocity location.)*

ERROR 1501 Problem with processing program arguments *(A fatal problem occured when command line options were read. The help /section 4.4/ should be consulted.)*

ERROR 1502 Trifile *filename* cannot be read

ERROR 1503 Incorrectly placed meshlines *(The meshlines read from the meshfile do not correspond with the geometry read from the trifile.)*

ERROR 1504 No patches generated in box *box* despite the fact there are different in/out statuses *(The box has different in/out statuses; therefore, some boundary faces must be generated here. This problem might be caused by unclean geometry near the box.)*

ERROR 1505 Tying patches routine: cannot break the required edge *(A logical error occured in routine of tying patches.)*

ERROR 1506 Tying patches routine: cannot move the required node *(A logical error occured in routine of tying patches.)*

ERROR 1507 The closed loop cannot be gathered for box *X* (ii=*I*, jj=*J*, kk=*K*, bx==*box*) *(This error is the most common symptom of problematic topology of input triangles. Check the triangulated surface in the global box I,J,K.)*

ERROR 1508 Processing of command line: cannot find any cell index for the switch -viewc *(A fatal problem occured when command line option -viewc was processed. The help /section 4.4/ should be consulted.)*

ERROR 1509 Processing of command line: cannot find correct number of parameters for the switch -viewcgrp *(A fatal problem occured when command line option -viewcgrp was processed. The help /section 4.4/ should be consulted.)*

ERROR 1510 Processing of command line: cannot find any boundary number for the switch -viewb *(A fatal problem occured when command line option -viewb was processed. The help /section 4.4/ should be consulted.)*

ERROR 1511 Unknown switch *switch* encountered *(An unknown switch was encountered when processing the command line)*

ERROR 1512 No input ascii file was specified *(The meshing task is expected; however, the name of the input meshfile was not specified)*

ERROR 1513 Identical triangles have been detected in the trifile *filename*: triangles *T1* and *T2* are identical

ERROR 1514 A flaw in triangle topology was detected in trifile *filename*

ERROR 1515 Triangle *T* has an unstitched edge in trifile *filename (Repair the triangulated surface in the preprocessor first.)*

ERROR 1516 Neighbouring triangles have opposite orientation (*T1* and *T2*) *(The orientation of triangles should be harmonized in the preprocessor first.)*

ERROR 1519 Processing of command line: -viewijk needs to be followed by six integer values *(A fatal problem occured when command line option -viewijk was processed. The help /section 4.4/ should be consulted.)*

ERROR 1520 Invalid mesh size and/or placement *(No boxes were generated. The placement of meshlines should be checked in the preprocessor.)*

ERROR 1521 Cannot write changed grid to file *X (It is not possible to write down the gridfile changed by routine making boundaries of two complementar grids conformal)*

ERROR 1522 Velocity location *X* neighbours to a box with incorrect cell index *X (This is an internal logic error; the system of generated boxes seems to be corrupted.)*

ERROR 1523 The modified surface cannot be written to file *X* *(It is not possible to write down the file X)*

ERROR 1524 The common face *X* has less than three vertices (*X* vertices) *(Check the triangulated surface in the global box I,J,K)*

ERROR 1525 The boundary face *X* has less than three vertices (*X* vertices) *(Check the triangulated surface in the global box I,J,K)*

ERROR 1526 The IP face *X* has less than three vertices (*X* vertices) *(Check the triangulated surface in the global box I,J,K)*

ERROR 1527 The cell *X* has less than four faces (*X* faces) *(Check the triangulated surface in the global box I,J,K)*

ERROR 1528 Incorrect patches were generated in box *X*. Check input geometry near triangles *X* and *X* *(Check the triangulated surface in the global box I,J,K)*

WARNING 1050 IN/OUT test seems to be uncertain (the confidence is *percent*%) *The in/out test has been evaluated; however, not all released testing rays had shown the same result.*

WARNING 1051 IN/OUT test of tested side seems to be uncertain (the confidence is *percent*%) *The in/out test has been evaluated; however, not all released testing rays had shown the same result.*

WARNING 1052 IN/OUT test seems to be uncertain *The in/out test has been evaluated; however, not all released testing rays had shown the same result.*

WARNING 1053 IN/OUT test of tested side seems to be uncertain (the confidence is *percent*%) *The in/out test has been evaluated; however, not all released testing rays had shown the same result.*

WARNING 1800 *N* of overlapped triangles were detected *This warning occurs when the rapid test of overlapping triangles finds any overlap. For details, see sections 4.5 and 4.7.*

WARNING 1801 Not all faulty triangles have been repaired *The routine for automatic reparation of triangles was not able to repair all problems*

WARNING 1802 Boundary refinement information is set in trifile and also in meshfile *(It is not clear which type of definition of boundary refinement should be read, because both types were used. The boundary refinement which was set in the trifile will be ignored in this case. The user should check what was his intent. For more information about setting up the refinement, see section 4.6.)*

WARNING 1850 There are no patches on the first geometry that can be matched to non-conformal patches of the second geometry; please check the list of complementary boundaries *(It seems that lists of boundaries that form the interface between two complementary boundaries are not correct.)*

WARNING 1851 There are no patches on the second geometry that can be matched to non-conformal patches of the second geometry; please check the list of complementary boundaries *(It seems that lists of boundaries that form the interface between two complementary boundaries are not correct.)*

WARNING 1852 There are patches which have no possible patch to make a pair (X in GEO #1 and X in GEO #2). *(There are patches that cannot be matched with any other patch. The lists of boundaries that form the interface between two complementary boundaries should be checked.)*

4.10 Grid Data Structure

4.10.1 Grid components

In VECTIS-MAX , both the grid connectivity and geometrical data are defined and maintained for the single computational mesh, generated over multiple fluid and solid domains. The global domain is composed of fluid and solid domains. The computational mesh is obtained by dividing the global domain in a number of non–overlapping finite *control volumes* (CV) or cells which constitute the computational mesh.

The computational mesh can be described in terms of (see Figure 4.12, left):

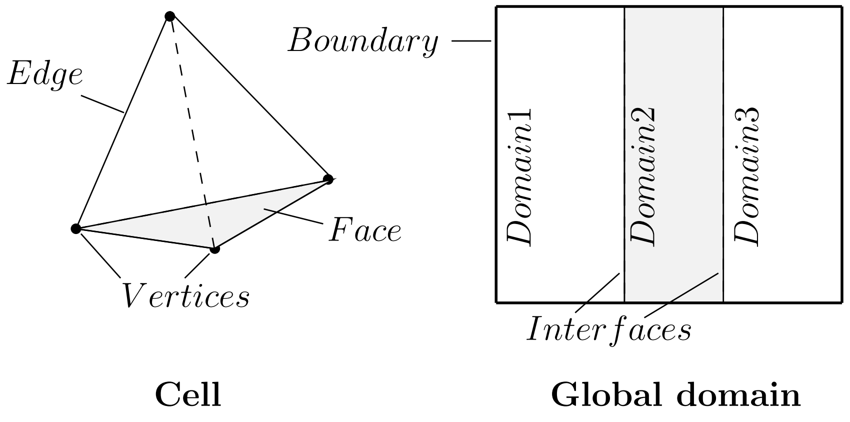


Figure 4.12: Grid objects

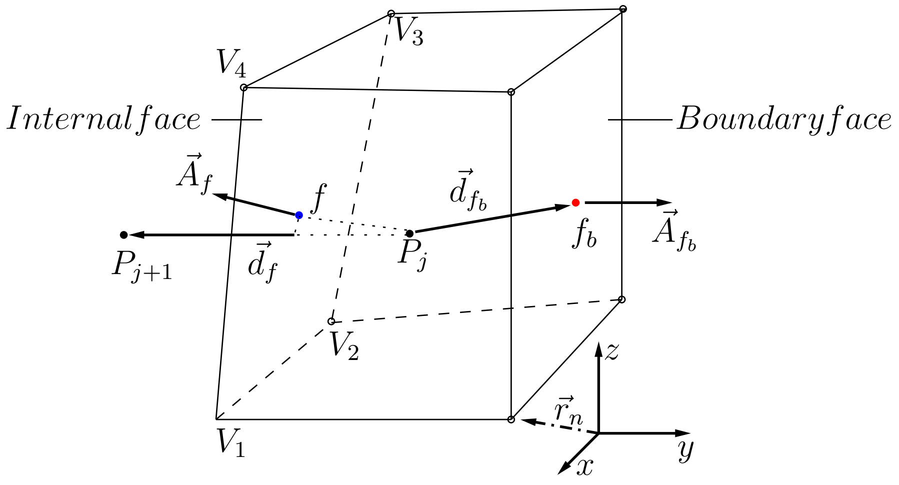


Figure 4.13: Control volume and notation

- *vertices (nodes)*
- *edges*
- *faces* and
- *cells*

Each of these grid objects need to have an index (label) and need to be defined in terms of other objects (with exception of vertices). Information provided in order to identify and connect each grid object adjacent to the given object are commonly called *connectivity data*.

Vertices ($V_n,\ n = 1, N_v$) are basic grid objects since they define physically the numerical grid through their position vectors $\vec{r}_n(x_k,t)$, where $x_k, k = 1,2,3$ are Cartesian coordinates. The position vectors are defined with respect to a (fixed) Cartesian coordinate system (x,y,z) as it is shown in Figure 14.1, which illustrates a general control volume. An *edge* is a staright line which is formed by connecting two neighbouring vertices. The line connecting vertices V_1 and V_2 in Figure 4.14 represents and edge. A *face* is defined in terms of vertices and forms a plane in 3D. In 2D, a face is simply an edge. In Figure 14.1 a face is formed by connecting vertices V_1, V_2, V_3 and V_4. A *cell* is defined by a number of faces that forms a closed volume in 3D. A number of cell shapes are given in Figure 4.14.

A face which is shared by two adjacent cells is an *internal face*, otherwise it is a *boundary face*. Thus, a face is usually almost planar, simple connected polygon, which can be uniquely defined by the corresponding list of vertices. In this list, the face vertices are ordered to obey the right–hand orientation with respect to the face normal vector $\vec{A}_f$. In other words, the vertices are ordered in the counter–clockwise direction when seen from the direction which is opposite to the face normal. In VECTIS, the face normal is always directed from a cell with a lower index to the cell with a higher index (with reference to the above figure, $j < j+1$ for the face f). In case of boundary faces, normals always point out of the adjacent cell.

The building blocks of the computational domain are:

- *global domain*
- *fluid/solid domains*
- *interfaces* and
- *boundaries*

The *global domain* in most cases coincides with the computational domain and it can have any number of fluid/solid domain (Figure 4.12, right). The *fluid domain* is made of a single fluid material domain whereas the *solid domain* can be made of one or more materials. A fluid domain can be further sub-divided into a fluid sub-domain to include additional fluid transport equations. *Boundaries* represent the external boundaries of the global domain as in Figure 4.12 (right). These are sub–divided into non–overlapping boundary regions according to the boundary condition type. *Interface regions* are implicitly defined at fluid/solid and solid/solid material interfaces (see Figure 4.12, right). For every pair of adjacent material domains there is one interface region.

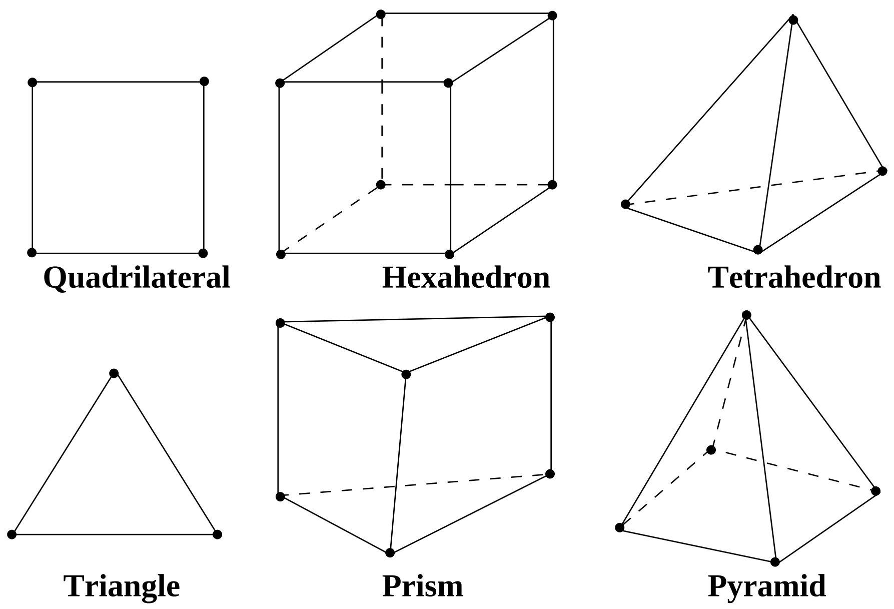


Figure 4.14: Various 2D and 3D cells (CVs)

4.10.2 Calculation of grid geometric properties

Geometric properties of cells and faces play a vital role in the discretisation procedure. For M_v properly ordered face vertices, Figure 14.1, the face surface vector $\vec{A}_f$ is computed by using formulae for a triangular face $\vec{A}_t$ since each polygon can be decomposed into $M_t = (M_v - 2)$ triangles. Thus, the following expression is used:

$$\vec{A}_f = \sum_{t=1}^{M_t} \vec{A}_t = \frac{1}{2} \sum_{i=3}^{M_v} [(\vec{r}_{i-1} - \vec{r}_1) \times (\vec{r}_i - \vec{r}_1)] \tag{4.1}$$

Note that the use of the above equation ensures the geometric conservation law:

$$\oint_A d\vec{A} = \sum_{f=1}^{n_f} \vec{A}_f = 0 \,, \tag{4.2}$$

where n_f is the number of cell–faces. The cell face area is then calculated as:

$$A_f = \sum_{t=1}^{M_t} |\vec{A}_t| = \sum_{t=1}^{M_t} \sqrt{(A_{tx})^2 + (A_{ty})^2 + (A_{tz})^2} \tag{4.3}$$

The coordinates of the face centre $\vec{r}_f = \left(x_j \,, y_j \,, z_j\right)$ are computed as the average values of the centre coordinates of decomposed triangles $\vec{r}_t, t = 1, M_t$, weighted by their areas $A_t = |\vec{A}_t|$:

$$\vec{r}_f = \left(\sum_{t=1}^{M_t} A_t \vec{r}_t \right) / \left(\sum_{t=1}^{M_t} A_t \right) \tag{4.4}$$

The face centre coordinates of a triangle are simply average values of its vertex coordinates.

The cell volume formula is based on the Gauss' theorem:

$$\int_V \nabla \vec{r}\, dV = \oint_A \vec{r} d\vec{A} \rightarrow V_{P_j} = \frac{1}{3}\sum_{f=1}^{n_f} \vec{r}_j \cdot \vec{A}_f = \frac{1}{3}\sum_{f=1}^{n_f} \left(\sum_{t=1}^{M_t(f)} \vec{r}_{tf} \cdot \vec{A}_{tf} \right). \tag{4.5}$$

The geometric centre of a control volume around P_j is defined as:

$$\vec{r}_{P_j} = \frac{1}{V_{P_j}} \int_V \vec{r} dV \tag{4.6}$$

With the help of Gauss' theorem and by using the property of a planar face $\vec{r} \cdot \vec{n} = const$ the following equation for the position vector at the cell centroid can be derived:

$$\vec{r}_{P_j} = \frac{1}{4V_{P_j}} \sum_{f=1}^{n_f} \left(\vec{r}_f \cdot \vec{A}_f \right) \vec{r}_f = \frac{1}{4V_{P_j}} \sum_{f=1}^{n_f} \left[\sum_{t=1}^{M_t(f)} \left(\vec{r}_{tf} \cdot \vec{A}_{tf} \right) \vec{r}_{tf} \right]. \tag{4.7}$$

The interpolation factor defined in terms of distances between the node P_j, the centre of face f and P_{j+1} (see Figure 14.1) is often used when the cell face values are obtained by linear interpolation:

$$w_f = \frac{|\vec{r}_{P_{j+1}} - \vec{r}_f|}{|\vec{r}_f - \vec{r}_{P_j}| + |\vec{r}_{P_{j+1}} - \vec{r}_f|} \tag{4.8}$$

The distance vectors for internal ($\vec{d}_f$), boundary faces ($\vec{d}_{f_b}$) and faces along material interfaces ($\vec{d}_i$) are also required. They are defined as:

$$\vec{d}_f = \vec{r}_{P_{j+1}} - \vec{r}_{P_j}\ ,\ \vec{d}_{b_f} = \vec{r}_{b_f} - \vec{r}_{P_j}\ ,\ \vec{d}_i = \vec{r}_i - \vec{r}_{P_j} \tag{4.9}$$

where $\vec{r}_{b_f}$ and $\vec{r}_i$ are the position vectors at the centres of the boundary face and interface, respectively. Finally, normal distances between centres of near-boundary cells and adjacent boundary faces are used to implement various boundary conditions. They are defined as:

$$\delta_n = \left(\vec{A}_{f_b} \cdot \vec{d}_{f_b} \right) / |\vec{A}_{f_b}| \tag{4.10}$$

Note that the angle between the distance vector $\vec{d}_f$ and the face area vector $\vec{A}_f$ is a measure of the grid non-orthogonality.

4.11 Mesh Import

4.11.1 Mesh requirement

The Vectis solver requires face–to–node and face–to–cell connections (often called face addressing) and also cell–to–face connections. The connectivity data can be then described by:

- □ Number of vertices, number of internal and boundary faces, and number of cells.

- ◻ List of coordinates for all vertices.
- ◻ List of vertices that form a face – for all faces (face–to–node connectivity).
- ◻ List of two cells (for a boundary face one cell) adjacent to a face – for all faces (face–to–cell connectivity).
- ◻ List of faces that enclose a cell – for all cells (cell–to–face connectivity)

4.11.2 Importing a grid data file

The grid connectivity, defined above, is the one used inside the VECTIS-MAX solver. Obviously, it does not provide association of grid data with other data structure objects such as domains and boundary or interface regions. The import of a grid file is done by reading in the appropriate grid file, *project_name.GRD*. The VECTIS-MAX grid file format .GRD is generated using VECTIS-MAX mesher “vmesh”. The following file formats are supported in VECTIS-MAX solver:

- ◻ VECTIS-MAX grid file (.GRD)
- ◻ Universal grid file (.unv)
- ◻ Vectis 3 (.DAT)
- ◻ Nastran (.nas)
- ◻ Cedre (.ccm)
- ◻ Spider (.flma)
- ◻ CGNS (.cgns)

When the vsolve command is run without specifing a grid file name, a message appears advising the user of accepted grid file types:

```
[1] linxx ~% vsolve
 USAGE: vsolve [-np 2] [-grid #type] [-debug #level] proj_name
        where #type =
                       0 (GRD) - Default
                       1 (unv) - Universal
                       2 (DAT) - Vectis-3
                       3 (nas) - Nastran
                       4 (ccm) - Star CCM+
                       5 (flma)- Spider
                       6 (cgns)- CGNS
[2] linxx ~%
```

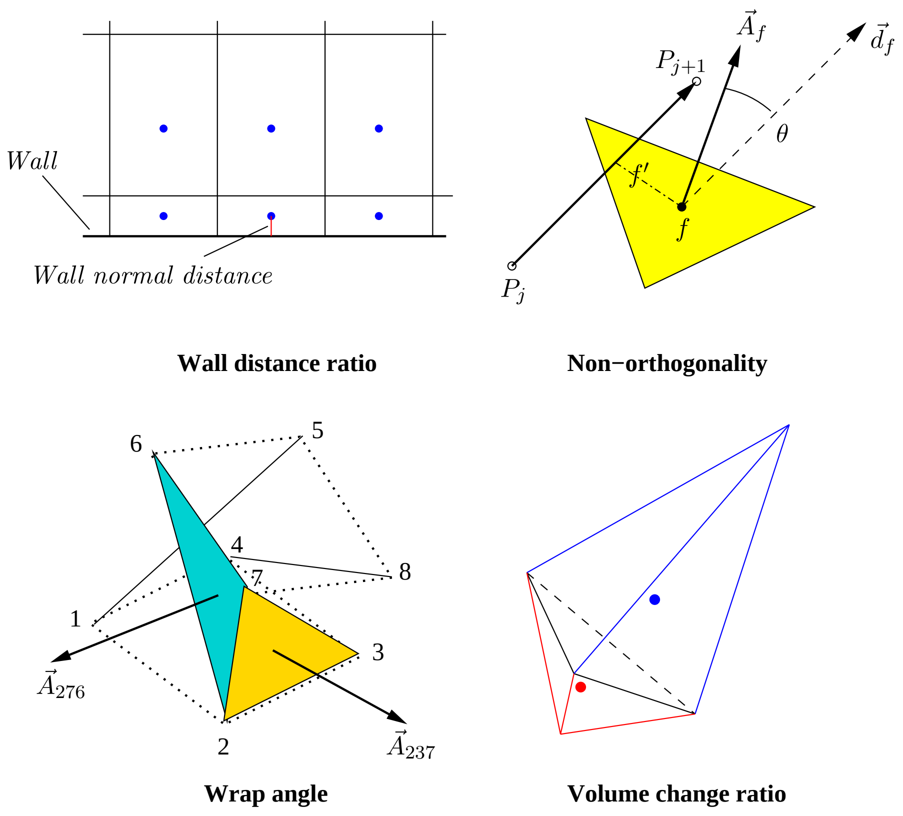


Figure 4.15: Mesh quality checks

4.12 Mesh Quality Checks

The finite volume discretisation deals with cells of any shape such as hexahedra, prisms, tetrahedra, pyramids and Vectis cut-Cartesian cells. However, in terms of the numerical accuracy and stability the body-fitted hexahedral or prismatic grids near boundaries are the best choice. The quality of the mesh used in a CFD simulation is very important. There are a number of factors that affect stability and numerical accuracy. A number of criteria can be used to assess the quality of a computational mesh. The following checks are performed in VECTIS-MAX (see Figure 4.15 for mesh checks 3, 4 ,5 and 6):

1. *Negative cell volume*: If this is the case then the user is warned and the appropriate warning message displayed.
2. *Cell enclosure*: If the sum of cell-face vectors for each cell is not zero or very close to zero (10^{-10}) then the cell is not completely enclosed by its faces.
3. *Wall distance ratio for near–wall cells*: it represents the ratio of the wall normal distance and an average normal distance of cell neighbours to the same wall.
4. *Non-orthogonality*: angle between the face vector $\vec{A}_f$ and the distance vector $\vec{d}_f$ connecting two neighbour cells, ≤ 75. This introduces the cross–diffusion term and modifies the discretisation of the diffusion.
5. *Warp angle*: angle between surface normals of the triangulated cell-faces, ≤ 50.

6. *Volume change ratio*: ratio between a cell volume and the maximum volume among its neighbours ≤ 2.

Criteria 3, 4 and 6 are used to identify poor quality cells. The user is informed of the total number of poor quality cells:

```
*****Checking quality of cells: there are    20 poor quality cells
```

The way such cells are treated makes the subject of section Poor Quality Cell Treatment.

5 READING & MANIPULATING MESHES

5.1 Introduction

Vpre is required to be run after the mesh (grid) file is generated and prior to the running of the solver (vsolve). The main argument is the grid file in addition to various other arguments,

```
vpre [options] fluid.GRD
```

Vpre automatically performs optimal cell reordering (Reverse Cuthill-McKee, if grid file is unordered). Optional arguments to vpre are as follows:

-inp #file	Input-file (boundary definition), same input file supplied to vsolve.
-metis #meth	Mesh partitioning method (METIS) (see Section (5.2)).
-np #num	Number of partitions for parallel run.
-rest	Repartition restart files along with mesh files (see Section (5.4)).
-agi #num	Define Arbitrary Grid Interface boundaries ((see Section (5.3))).
-export #form	Export boundary mesh to specified format, where #form can be one of unv (Universal), dx (OpenDX), gnu (gnuplot) or x3D.
-nomat	Do not equally divide materials during decomposition of a multi-material mesh. (see Section (5.2)).
-jtype #num	Mesh joining method ((see Section (5.3))).
-o #file	Output file name when joining meshes (overrides default "COALESCED.GRD").

Controlling environment variables:

VECPRE_REORDER_MATERIAL	Set to OFF to disable material reordering
AGI_EXTRUDE	Specify which agi boundary to extrude during extrusion process. Set to -1 to indicate user interaction (if more than one agi boundary to specify).

5.2 Mesh Partitioning

In order to run the solver (vsolve) in parallel on multiple processors, it is first necessary to decompose (partition) the single mesh file into the corresponding parallel files. Mesh partitioning is controlled via the "-np" and (optionally) the "-metis" argument. For example:

```
  vpre -np 4 -metis rb test.GRD
```

will create 4 new mesh files in the corresponding directories P001, P002 etc. using the "recursive bisection" method. There are 3 possible partitioning methods (all METIS based):

-rb	(Recursive Bisection)
-kway	(K-way)
-vkway	(VK-way)

In general, the default partitioning method used is recommended. By default, for np < 8, the "recursive bisection" method is used, otherwise the "kway" method is selected. For further information visit the site http://glaros.dtc.umn.edu/gkhome/views/metis. If the grid file contains more than one material, then METIS will, by default, also attempt to equally divide the materials amongst the partitions. This behaviour can be disabled via the "-nomat" option.

In addition to the automatic mesh partitioning, the user can manually define mesh cut planes via an input file ("PRECUT.DEF"). This file is used only if it is found in the working directory. The typical format within is:

```
PLANE ORIGIN=0.,-0.15,0. NORMAL=0,1,0
PLANE ORIGIN=0.,-1.15,0. NORMAL=0,1,0
PLANE ORIGIN=0.,-2.15,0. NORMAL=0,1,0
```

This example defines 3 plane cuts through a mesh. If the number of partitions resulting from the user defined planes was less than the number of partitions required, then further decomposition is performed on each user defined partition.

Additional automatic checks are performed by "vpre". These include checking the partition and material interface boundaries do not coincide in the case of multi-material grid files. If they do, the partition boundaries are adjusted. This is done by looping over partitions in order of size (so as to minimize the degree of load-imbalancing), and expanding these, where necessary, across material interfaces.

In order to improve pre-conditioning in "vsolve", each partition, in decreasing order of neighbourliness, is coloured according to rule: no two colours are allowed to be neighbours. The partition numbers are then renumbered according to this colouring. For example:

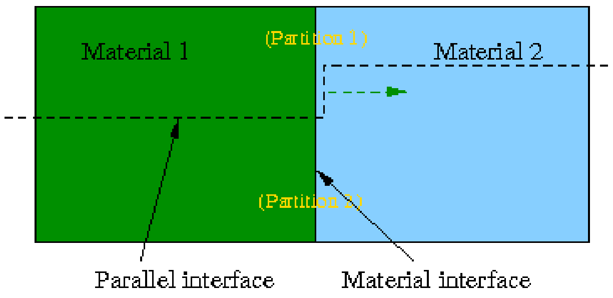


Figure 5.1: Picture of 2 materials/partitions with slight coincidence. Green arrow shows how partition interface is moved from smaller (1) to larger partition (2).

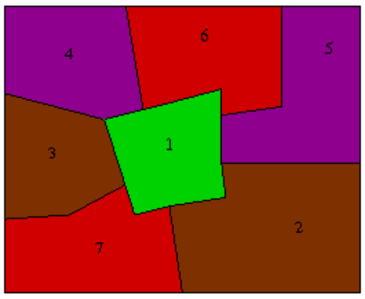


Figure 5.2: Example of colouring for 7 partitions.

5.3 Mesh Joining

This section describes the way in which a multi-material grid file can be created from coalescing separate single material grid files via the use of *vpre* . The use of the acronym AGI (Arbitrary Grid Interface) below denotes boundary regions to be joined.
Prior joining the meshes, it is important to generate them in the correct way, i.e. the AGI boundaries should be specified when *vmesh* is run, see Sec (4.8) for a detailed explanation. This is to ensure the best possible join. There are currently five ways of joining separate grids via the -jtype option. The following command:

```
vpre -jtype 0 fluid1.GRD solid1.GRD
```

would create a 2-domain grid (fluid & solid) from the 2 single domain grids. It is important that the fluid grid files are listed FIRST on the command line. The preferred method of joining is "jtype-0" where the input meshes have near (or total) conformal boundaries. An example of creating a 3-material grid file beginning from the mesh creation stage is shown below:

The middle mesh (solid1) in Fig (5.3) represents a solid domain, the two surrounding meshes are fluid domains. The coincident boundaries to be joined are fluid1(boundary 1)-to-solid1(boundary 2) and fluid2(boundary 1)-to-solid1(boundary 3). The three meshes are initially created using the "-int" option to specify the boundaries to joined (and subsequently converted to interface regions):

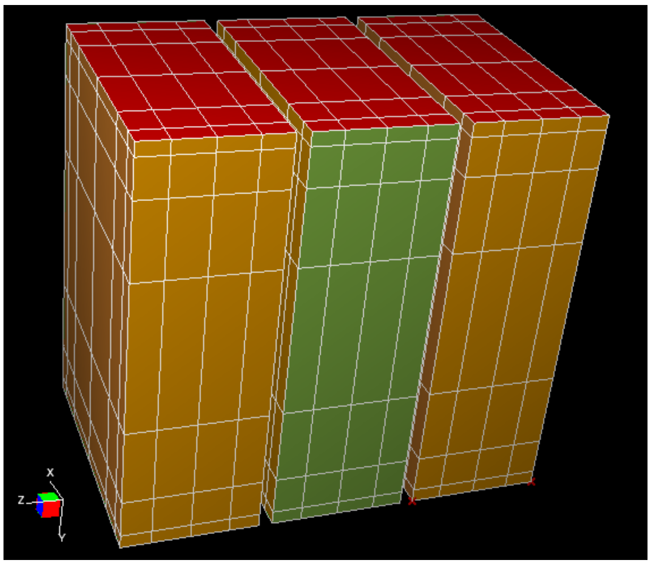


Figure 5.3: Picture of three meshes to be joined. Gaps between meshes are exaggerated.

```
vmesh -int 1 fluid1.mesh
vmesh -int 1 fluid2.mesh
vmesh -int 2 3 solid1.mesh
```

Next, additional mesh processing is preformed using the ("-conformrew" option). This is to ensure these boundary regions are made as conformal as possible. These commands are repeated for each pair of meshes to be joined.

```
vmesh -conformrew fluid1.GRD 1 solid1.GRD 2
vmesh -conformrew fluid2.GRD 1 solid1.GRD 3
```

Note, as an alternative to the above:

```
vmesh -conform fluid1.GRD 1 solid1.GRD 2
vmesh -conform fluid2.GRD 1 solid1_conform.GRD 3
```

The first "-conform" would produce two new files "fluid1_conform.GRD" and "solid1_conform.GRD". The next "-conform" would create the files: "fluid1_conform.GRD" and "solid1_conform_conform.GRD". This method is safer (i.e. doesn't overwrite original meshes) but a little less convenient to type.

For the final stage, the three meshes are joined using the *vpre* utility (remembering to place the fluid domains first).

```
vpre -jtype 0 fluid1.GRD fluid2.GRD solid1.GRD
```

This would produce a multi-material file. By default this is called COALESCED.GRD, unless the

"-o filename" option is specified. All the boundary faces that are successfully joined are converted to interfaces (internal faces along a material interface region). Any boundary faces not joined are stored in the grid file as boundaries. When this multi-domain grid is imported into *R–Desk* , these unjoined regions will appear as with boundary type: "Unjoined Boundary".

The full listing of all the jtype options are shown in Table (5.1). The options above "0" are in general less reliable/robust and should in general be treated by the user as experimental.

5.3.1 Sub-domain joining

Certain physical models such as porous media, fan and radiator require creation of sub-domains. Since any fluid or solid material domain can contain any number of non-overlapping sub-domains the surface mesh along sub-domain interfaces should be conformal.

In order to perform a conformal joining, firstly *vmesh* is used to make interfaces conformal in a similar way described above.

Consider a catalyst test case consisting of a single domain (air_no_cat) that should be joined with 2 sub-domains (cat1 & cat2). Furthermore, interfaces 7 & 8 of the domain should be made conformal to interfaces 2 & 3 of the sub-domains. To make these interfaces conformal the following *vmesh* commands are issued:

```
vmesh -int 7 8 air_no_cat.mesh
vmesh -int 2 cat1.mesh
vmesh -int 3 cat2.mesh
vmesh -conformrew cat1.GRD 2 air_no_cat.GRD 7
vmesh -conformrew cat2.GRD 3 air_no_cat.GRD 8
```

Then *jtype* option is used in *vpre* to perform the joining as follows:

```
vpre -no-material-offset -jtype 0 air_no_cat.GRD -subdom cat1.GRD -subdom cat2.GRD -o air_cat
```

where -*subdom* is used to indicate that the next mesh file is a sub-domain of the preceding mesh (parent); -*no-material-offset* ensures that offsetting of material id's (with the same id) is not allowed when reading multiple grids. The generated grid file *air_cat.GRD* now contains both sub-domain grids.

5.4 Restarting On A Different Number Of Partitions

In order to stop/restart the solver (vsolve) on a different number of processors, it is first necessary to run vpre with the "-np" and "-rest" argument in order to repartition the grid and restart files concomitantly. For example, in order to switch from 4 to 8 processors:

```
vpre -rest test.rst0_010 -np 8 flow.GRD
```

JTYPE	USAGE
0	Conformal join. This method is the simplest and should be used when the input meshes are known to be conformal (i.e. boundary faces are the same along AGI boundaries). Any non-conformal faces are left unconnected and remain as boundary faces.
1	This method should be used when there is an initial gap between 2 input meshes. This gap is closed with a tetrahedral/conformal mesh. Typically the 2 input meshes would be 2 fluid domains and the gap would represent the solid domain. Any new boundary faces generated are all painted with the same boundary number.
2	This method is appropriate when the boundaries to be joined on each input mesh are nearly coincident, but not necessarily conformal, and thus require mesh extrusion on one side in order to create a gap to be re-meshed tetrahedrally (as in jtype-1). The choice of which mesh to extrude is, by default, the mesh containing the larger (on average) cells along the AGI interface. This can be overridden via the AGI_EXTRUDE environment variable. Any new boundary faces should inherit the boundary numbers from those faces removed (as a result of extrusion).
3	Same as jtype-2 except mesh is extruded both sides of the AGI interface.
4	Similar to jtype-0 except any unconnected faces are then dealt with using the jtype-2 extrusion/re-mesh approach. This method is appropriate when the interface regions are known to be nearly conformal.

Table 5.1: Table of all the -jtype options.

This would generate 8 grid files in the directories P001, P002 etc. along with the corresponding 8 restart files (P001test.rst0_010 etc.). Vpre expects the parallel (corresponding to np=4) grid and restart files from the previous run to be present in the parallel directories (P001 etc.). If no filename argument is passed to "-rest" then the latest restart files corresponding to the grid file name (e.g. P001flow.rst1_010 etc. in this case) are used.

6 SOLVER FUNDAMENTALS

6.1 Introduction

The solver is a software module which performs the numerical simulation of a CFD/Continuum Mechanics problem. In other words, it produces the numerical results which simulate transport of mass, momentum, energy and other associated physical phenomena in the considered *continuum*. Continuum is either fluid or solid material whose behaviour is governed by transport equations expressing basic conservation laws of physics for

- Mass,
- Momentum (Newton's second law),
- Energy (First law of thermodynamics) and
- Entropy (Second law of thermodynamics).

The spatial domain, occupied by the fluid or solid continuum, is defined as *material domain* if the continuum does not exchange mass with other material domains. Material domains are separated from the external environment by domain *boundaries* and from other material domains by domain *interfaces*.

Our knowledge about physical conditions at domain boundaries and interfaces should determine the behaviour of the continuum within a material domain. For the time–dependent problems, the continuum behaviour will also depend on the *initial conditions*. Considering an individual transport equation, so–called *numerical boundary/interface conditions* need to be imposed at domain boundaries and interfaces. These are *fixed* variable values (Dirichlet conditions), zero or specified gradient values (Neumann conditions) and mixed (Robin) conditions. At the continuum level, it is useful to introduce *physical boundary or interface conditions* along boundaries and interfaces. Their role is to define and if necessary update, in a physically correct way, the numerical boundary conditions for each transport equation. Typical examples of physical boundary conditions are walls, symmetry planes, flow inlets and flow outlets.

In practice, CFD problems are usually characterised by complex physics and complex geometry, involving a number of fluid/solid materials and associated material domains. This means that we have to model both geometry (replace real spatial domains with the material domains) and physics. After modelling, a numerical method is employed to solve the governing equations. The numerical method converts the mathematical model into a system of algebraic equations through the discretisation process. This process involves the discretisation of

□ spatial domains, i.e. space,

□ time

□ and transport equations.

The space discretisation creates a numerical mesh which describes the solution domain in terms of a finite set of non–overlapping *control volumes* or *cells*, enclosed by *faces* and supported by *vertices*. With the mesh support, the governing equations are discretised over each cell.

In summary, the numerical simulation comprises the following basic steps:

CFD PROBLEM DEFINITION $\Leftarrow$ Real world described by

□ Spatial domains (geometry) & their boundaries/interfaces

□ Conservation laws

□ Continuum properties

MATHEMATICAL MODELS $\Leftarrow$ Modelling real world

□ Solution (computational) domain & physical boundary or interface conditions

□ Governing equations

□ Constitutive and thermodynamic relationships, thermo-physical properties

NUMERICAL METHOD $\Leftarrow$ Transforming models into a system of algebraic equations

□ Spatial (meshing) & time discretisation

□ Discretisation of modelling equations

□ Initial and numerical boundary conditions

□ Solution algorithm (linearisation & linear equation solver)

NUMERICAL RESULTS $\Rightarrow$ Post–processing

In what follows, the fundamental mathematical models that underpin the solver design are presented:

□ conservation equations,

- their closure problem,
- Reynolds and Favre averaging which leads to the
- Reynolds averaged equations.

Other solver related chapters describe modelling of

- Spatial domains
- Continuum (material) properties
- Turbulence
- Single–phase flows and
- Multi–phase flows

The next two chapters are devoted to the implementation and setting of Boundary Conditions and to the Numerical Solution.

6.2 Continuum Conservation Equations

Before presenting basic conservation equations, describing both single–phase and multi–phase flow, several terms relevant to continua and multi–phase flows are introduced.

6.2.1 Terminology

Continuum The term *continuum* is used to describe the matter which is continuously distributed in space. Mathematically, properties of continuum such as pressure, density, temperature and velocity are defined as single–valued continuous functions of time and space.

Phase The physical state of continuum can be either fluid (gas or liquid) or solid, and a *phase* is the thermodynamic definition for these states of matter, see for example Moran and Shapiro [1992]. More precisely, the term phase refers to homogeneity in both physical structure of matter and chemical composition. A phase can consist of one or more *chemical components*. For example, gases can be mixed in any proportion to make a single gas phase. Only certain liquids (e.g. alcohol and water) can be mixed to form a single liquid phase while other such as oil and water are not miscible and form two liquid phases. Note that in some cases the concept of a phase used in CFD can have a broader sense than the one used in thermodynamics.

Pure substance Another thermodynamics term is a *pure substance* which contains one or more chemical components with uniform and fixed chemical composition. An example is Air, which is the mixture of Nitrogen, Oxygen and other gases. Note that a pure substance can exist in more than one phase; for instance liquid water and water vapour form two phases.

Multi–component phase A phase which has more than one pure substance will be called the *multi-component* phase.

Species Here and generally in CFD, a fluid phase component is often referred as *species*, having the same meaning as the pure (compressible) substance.

6.2.2 Instantaneous equations

The global laws of continuum physics: the conservation of mass, momentum, and energy are usually applied to a certain spatial region – control volume (CV) rather than to a given mass of continuum. The former is known as the *Eulerian* or *control volume approach* while the latter is the *Lagrangian* approach. For the infinitesimally small control volume the conservation equations can be transformed into either a differential coordinate-free form or a form specific to a chosen coordinate system. The VECTIS-MAX solver employs a Cartesian coordinate system (x,y,z). Throughout this manual a compact tensor notation is often used. Accordingly, if a monomial contains a repeated index a summation over the range of index values is applied (Einstein convention). A free or un-repeated index is the one which appears once in a monomial. Two free indices define the second order tensor, one free index defines vector, while scalar does not have a free index.

Let us consider an arbitrary control volume V, bounded by generally moving surface $\vec{A}$ whose velocity is $\vec{U}_g$. In this case (a moving grid) it is convenient to formulate the conservation equations in a moving frame of reference. This formulation is usually referred to as *Arbitrary Lagrangian–Eulerian* (ALE) approach. When the CV–surface velocity $\vec{U}_g$ is zero (CV fixed in space) the Eulerian form of equations is obtained. When, however, the CV–surface moves with the fluid velocity vector $\vec{U}$ (i.e. $\vec{U}_g = \vec{U}$) the control volume becomes identical to the control mass and the Lagrangian form of equations is recovered. Noticing that the instantaneous velocity vector $\widehat{U}_i$ and pressure $\widehat{p}$ as well as other flow quantities vary both with a position x_i and time t, both coordinate–free integral and Cartesian differential form of conservation equations are presented.

6.2.2.1 Mass Conservation

$$\frac{d}{dt}\int_V \widehat{\rho} dV + \oint_A \widehat{\rho}\left(\vec{\widehat{U}} - \vec{U}_g\right)\cdot d\vec{A} = 0 \tag{6.1}$$

$$\frac{\partial \widehat{\rho}}{\partial t} + \frac{\partial}{\partial x_j}\left[\widehat{\rho}\left(\widehat{U}_j - U_{g,j}\right)\right] = 0 \tag{6.2}$$

In the above equations $\widehat{\rho}$ represents density. If the continuum is a multi–component phase (a mixture of species) the mass conservation for each species i must be satisfied. This conservation is usually expressed in terms of the local concentration (mass fraction) $\widehat{c}_i$ which is the ratio of the mass of i–th species m_i to the total mass m of the mixture:

$$\widehat{c}_i = \frac{m_i}{m} \tag{6.3}$$

The corresponding transport equations are:

$$\frac{d}{dt}\int_V \widehat{\rho}\widehat{c}_i dV + \oint_A \widehat{\rho}\widehat{c}_i\left(\vec{\widehat{U}} - \vec{U}_g\right)\cdot d\vec{A} = \oint_A \vec{\widehat{J}}_{c_i}\cdot d\vec{A} + \int_V s_{c_i} dV \tag{6.4}$$

$$\frac{\partial}{\partial t}\left(\widehat{\rho}\widehat{c}_i\right)+\frac{\partial}{\partial x_j}\left[\widehat{\rho}\widehat{c}_i\left(\widehat{U}_j-U_{g,j}\right)\right]=\frac{\partial}{\partial x_j}\left(\widehat{J}_{c_i,j}\right)+\widehat{s}_{c_i} \tag{6.5}$$

where $\widehat{J}_{c_i,j}$ is diffusion flux of species and $\widehat{s}_{c_i}$ denotes the species source. The source can be the net rate of production per unit time and volume due to chemical reactions. If there is N_{sp} species then the definition of mass fractions implies:

$$\sum_{1}^{N_{sp}}\widehat{c}_i=1 \tag{6.6}$$

6.2.2.2 Momentum Conservation

$$\frac{d}{dt}\int_V\widehat{\rho}\vec{\widehat{U}}dV+\oint_A\widehat{\rho}\vec{\widehat{U}}\left(\vec{\widehat{U}}-\vec{U}_g\right)\cdot d\vec{A}=-\oint_A p\mathbf{I}\cdot d\vec{A}+\oint_A\widehat{\tau}\cdot d\vec{A}+\int_V\widehat{\rho}\vec{\widehat{f}}dV \tag{6.7}$$

$$\frac{\partial}{\partial t}\left(\widehat{\rho}\widehat{U}_i\right)+\frac{\partial}{\partial x_j}\left[\widehat{\rho}\widehat{U}_i\left(\widehat{U}_j-U_{g,j}\right)\right]=-\frac{\partial\widehat{p}}{\partial x_i}+\frac{\partial}{\partial x_j}\left(\widehat{\tau}_{ij}\right)+\widehat{\rho}\widehat{f}_i \tag{6.8}$$

where $\mathbf{I}$ is the unit tensor and $\widehat{\tau}=\widehat{\tau}_{ij}$ denotes the viscous stress tensor. One can notice that the Cauchy stress tensor

$$\widehat{\sigma}=-\widehat{p}\mathbf{I}+\widehat{\tau}\text{ or }\widehat{\sigma}_{ij}=-\widehat{p}\delta_{ij}+\widehat{\tau}_{ij} \tag{6.9}$$

is decomposed into the pressure and viscous stress tensor; δ_{ij} is is Kronecker symbol ($\delta_{ij}=1$ if $i=j$ and $\delta_{ij}=0$ otherwise). The gravitational body force and other external body forces are represented as $\widehat{f}_i$.

6.2.2.3 Energy Conservation

The First law of thermodynamics, applied to the given control volume, can be cast in the form of the total energy $\widehat{E}$:

$$\begin{aligned}&\frac{d}{dt}\int_V\widehat{\rho}\widehat{E}dV+\oint_A\widehat{\rho}\widehat{E}\left(\vec{\widehat{U}}-\vec{U}_g\right)\cdot d\vec{A}=\\&\oint_A\left(\vec{\widehat{q}}-\sum_{k=1}^{N_{sp}}\widehat{h}_k\vec{\widehat{J}}_{c_k}+\vec{\widehat{U}}\cdot\widehat{\sigma}\right)\cdot d\vec{A}+\int_V\left(\widehat{\rho}\vec{\widehat{f}}\cdot\vec{\widehat{U}}+\widehat{q}_v\right)dV\end{aligned} \tag{6.10}$$

$$\frac{\partial}{\partial t}\left(\widehat{\rho}\widehat{E}\right)+\frac{\partial}{\partial x_j}\left[\widehat{\rho}\widehat{E}\left(\widehat{U}_j-U_{g,j}\right)\right]=\frac{\partial}{\partial x_j}\left(\widehat{q}_j-\sum_{k=1}^{N_{sp}}\widehat{h}_k\widehat{J}_{c_k,j}+\widehat{U}_i\widehat{\sigma}_{ij}\right)+\widehat{\rho}\widehat{f}_i\widehat{U}_i+\widehat{q}_v \tag{6.11}$$

where the total energy is given as the sum of internal $\widehat{e}$, kinetic and other forms of energy:

$$\widehat{E}=\widehat{e}+\frac{\vec{\widehat{U}}^2}{2}+\text{other forms.} \tag{6.12}$$

Further, $\widehat{q}_j$ is the heat flux vector, $\widehat{h}_k$ represents the specific enthalpy of k–th species and $\widehat{q}_v$ is the energy source or sink term. The source term can include the energy source due to chemical reactions between species. The term involving specific enthalpies of species takes into account transport of different enthalpies by individual species.

The energy conservation expressed in terms of total energy is commonly used for incompressible flows. For compressible flows, the conservation of total enthalpy

$$\widehat{H} = \widehat{E} + \frac{\widehat{p}}{\widehat{\rho}} = \widehat{h} + \frac{\widehat{\vec{U}}^2}{2} \tag{6.13}$$

is the preferred form. The total enthalpy equation reads:

$$\begin{aligned} &\frac{d}{dt}\int_V \widehat{\rho}\widehat{H}dV + \oint_A \widehat{\rho}\widehat{H}\left(\widehat{\vec{U}} - \vec{U}_g\right)\cdot d\vec{A} = \\ &\frac{d}{dt}\int_V \widehat{p}dV + \oint_A \left(\widehat{\vec{q}} - \sum_{k=1}^{N_{sp}} \widehat{h}_k \widehat{\vec{J}}_{c_k} + \widehat{\tau}\cdot\widehat{\vec{U}} - \widehat{p}\vec{U}_g\right)\cdot d\vec{A} + \int_V \left(\widehat{\rho}\widehat{\vec{f}}\cdot\widehat{\vec{U}} + \widehat{q}_v\right)dV \end{aligned} \tag{6.14}$$

$$\frac{\partial}{\partial t}\left(\widehat{\rho}\widehat{H}\right) + \frac{\partial}{\partial x_j}\left[\widehat{\rho}\widehat{H}\left(\widehat{U}_j - U_{g,j}\right)\right] = \frac{\partial \widehat{p}}{\partial t} + \frac{\partial}{\partial x_j}\left(\widehat{q}_j - \sum_{k=1}^{N_{sp}} \widehat{h}_k \widehat{J}_{c_k,j} + \widehat{\tau}_{ij}\widehat{U}_i - pU_{g,j}\right) + \widehat{\rho}\widehat{f}_i\widehat{U}_i + \widehat{q}_v \tag{6.15}$$

6.2.2.4 Space conservation law

For control volumes with moving surfaces ($\vec{U}_g \neq 0$) the space (geometric) conservation law (SCL) should be satisfied:

$$\frac{d}{dt}\int_V dV - \oint_A \vec{U}_g \cdot d\vec{A} = 0 \tag{6.16}$$

If the above law is not enforced numerically the conservation of mass and other quantities is not guaranteed as artificial mass sources or sinks can be generated.

6.3 Closure Problem and Averaging

In order to solve the above instantaneous equations, they need to be closed by :

- □ Basic constitutive relations, specifying
  - Species diffusion fluxes $\widehat{J}_{c_k,j},\ k = 1, N_{sp}$
  - Viscous stress tensor $\widehat{\tau}_{ij}$
  - Heat flux vector $\widehat{q}_j$
- □ Equation of state, linking density and internal energy or enthalpy with basic thermodynamic variables: pressure $\widehat{p}$ and temperature $\widehat{T}$
- □ Thermo–physical properties associated with the constitutive relations and equation of state; for example dynamic viscosity, thermal conductivity, mass diffusion coefficient, density and specific heat.

In addition, well–posed boundary and initial conditions have to be specified. Note that the above conservation equations also describe the multi–phase flow, i.e. they describe the flow of each uncoupled phase. For this, interface conditions, providing the mass, momentum and energy balances at phase interfaces, are required.

However, due to a wide range of time and length scales associated with the *turbulence*, the current computer power enables the full numerical solution (Direct Numerical Simulation – DNS) of the closed instantaneous equations for very simplified flow cases.

A practical and well-established alternative to DNS is the solution of the mean–flow or *averaged equations* which are obtained from the instantaneous equations through some kind of the averaging or filtering process. The mean–flow equations require *turbulence models* in order to account for the turbulence effects rather than to simulate turbulence directly.

6.3.1 Reynolds and Favre averaging

Following classical Reynolds averaging, Hinze [1975], any instantaneous flow variable $\widehat{\phi}$ is decomposed into the mean and fluctuating part, respectively, as follows:

$$\widehat{\phi} = \overline{\phi} + \varphi' \tag{6.17}$$

Various averaging procedures can be used to define the mean. A general type of averaging is the *ensemble averaging* and it can be applied to any kind of flow. The ensemble average of random functions of space and time is defined as the arithmetic mean over many macroscopically identical realisations of $\widehat{\phi}(x_k,t)$, see for example Landahl and Mollo-Christensen [1986]:

$$\overline{\overline{\phi}}(x_k,t) = \lim_{M\to\infty} \frac{1}{M} \sum_{m=0}^{M} \widehat{\phi}^{(m)}(x_k,t) \ , \tag{6.18}$$

where $\overline{(\ldots)}$ represents the ensemble average, and $\widehat{\phi}^{(m)}(x_k,t)$ is the m-th realisation of $\widehat{\phi}(x_k,t)$. For the fluid flows, the macroscopically identical realisation means that statistically independent flows are exposed to the same set of initial and boundary conditions. Obviously the ensemble averaged quantity may be time dependent. In the statistically steady turbulence the ensemble average is the same as the time average:

$$\overline{\phi} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \widehat{\phi}(x_k,t)dt \tag{6.19}$$

In the case of a periodic flow with a period of T_p , it is convenient to use the *"periodic phase averaging"* over M_p periods:

$$\overline{\overline{\phi}}(x_k,t) = \overline{\overline{\phi}}(x_k,\vartheta) = \lim_{M_p\to\infty} \frac{1}{M_p} \sum_{m=0}^{M_p} \widehat{\phi}(x_k,t+mT_p) \ , \tag{6.20}$$

where t is now the instance corresponding to the particular periodic phase ϑ. The phase–averaged quantity is not averaged over the periodic phase but at the particular phase ϑ.

The following averaging rules apply, Tennekes and Lumley [1986]:

$$\begin{aligned} &\overline{\varphi'(x_k,t)} = 0 \, ; \overline{\varphi'\overline{\phi}} = 0 \\ &\overline{\widehat{\phi}\widehat{\psi}} = \overline{(\overline{\phi}+\varphi')(\overline{\psi}+\psi')} = \overline{\phi}\,\overline{\psi} + \overline{\varphi'\psi'} \ . \end{aligned} \tag{6.21}$$

Considering $\widehat{\phi} = \widehat{U}_i$, Equation (6.17) splits instantaneous motion, represented by the velocity field $\widehat{U}_i(x_k,t)$, into an organised (ensemble averaged) part $\overline{U}_i = \overline{\widehat{U}_i}$ and stochastic, turbulent part u'_i . The corresponding one point second moments $\overline{u'_iu'_j}$ (see the last term in Equation (6.21)) are rarely zero. They are known as the kinematic Reynolds stresses which are responsible for rates of momentum transport by the *turbulence*. Similar correlations are formed with other flow variables such as $\overline{\rho' u'_i}$, $\overline{c'_k u'_i}$ and $\overline{T' u'_i}$

In case of compressible flows, correlations involving fluctuating density and other variables are not negligible. A Favre or mass weighted average is then used to avoid the explicit modelling of such correlations. Compared to the Reynolds average, Equation (6.17), the alternative splitting is introduced:

$$\widehat{\phi} = \widetilde{\phi} + \varphi'' \tag{6.22}$$

where the Favre average $\widetilde{\phi}$ is defined as

$$\widetilde{\phi} = \frac{\overline{\widehat{\rho}\widehat{\phi}}}{\overline{\rho}} = \overline{\phi} + \frac{\overline{\rho'\varphi'}}{\overline{\rho}} = \overline{\phi} - \overline{\varphi''} \tag{6.23}$$

and the following relations hold:

$$\begin{aligned}
&\overline{\varphi'} = \widetilde{\varphi''} = 0, \;\; \overline{\widehat{\rho}\phi''} = 0, \;\; \widetilde{\varphi'} = -\overline{\varphi''} = \frac{\overline{\rho'\phi'}}{\overline{\rho}} \neq 0 \\
&\overline{\phi'\psi'} = \overline{\phi'\psi''} = \overline{\phi''\psi'} \\
&\widetilde{\phi''\psi''} = \frac{1}{\overline{\rho}}\overline{\widehat{\rho}\phi''\psi''} = \overline{\phi'\psi'} - \overline{\phi''}\;\overline{\psi''} + \frac{1}{\overline{\rho}}\;\overline{\rho'\phi'\psi'}
\end{aligned} \tag{6.24}$$

6.4 Reynolds–Averaged Equations

The classical Reynolds averaging is used in incompressible fluid mechanics while Favre averaging deals with compressible/reacting fluids where density fluctuations are significant. Applying Reynolds and Favre averaging to the instantaneous equations the incompressible and compressible forms of Reynolds Averaged Navier–Stokes (RANS) equations are obtained, respectively, see for example Speziale [1996]; Vandromme [1993]. While both forms are similar, a relation between Favre and Reynolds averages remains hidden unless the knowledge of Reynolds–averaged density fluctuation correlations is provided, see Veynante and Vervisch [2002].

The compressible RANS equations (Favre–averaged) are more complex then their incompressible counterparts. The Reynolds decomposition is used for density and pressure:

$$\widehat{\rho} = \overline{\rho} + \rho', \; \widehat{p} = \overline{p} + p' \tag{6.25}$$

while the Favre decomposition is applied to velocities, mass fractions, temperature, total enthalpy

$$\widehat{U}_i = \widetilde{U}_i + u''_i, \; \widehat{c}_i = \widetilde{c}_i + c''_i, \; \widehat{T} = \widetilde{T} + T'', \; \widehat{H} = \widetilde{H} + H'' \tag{6.26}$$

and other variables appearing in the convective terms. The Reynolds decomposition is generally employed for viscous stresses, mass diffusion and heat fluxes:

$$\widehat{\tau}_{ij} = \overline{\tau}_{ij} + \tau'_{ij}, \; \widehat{J}_{c_i,j} = \overline{J}_{c_i,j} + J'_{c_i,j}, \; \widehat{q}_j = \overline{q}_j + q'_j, \tag{6.27}$$

as there are no obvious advantages in using the Favre decomposition, Huang et al. [1995]. However, some care should be taken to satisfy that the average of fluctuating parts is zero, for example $\overline{\tau'}_{ij} = 0$. Employing the above combined Reynolds and Favre decompositions, ensemble averaging of the instantaneous equations results with the RANS equations:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_j}\left[\overline{\rho}\left(\widetilde{U}_j - U_{g,j}\right)\right] = 0 \tag{6.28}$$

$$\frac{\partial}{\partial t}\left(\overline{\rho}\widetilde{c}_i\right) + \frac{\partial}{\partial x_j}\left[\overline{\rho}\widetilde{c}_i\left(\widetilde{U}_j - U_{g,j}\right)\right] = \frac{\partial}{\partial x_j}\left(\overline{J}_{c_i,j} + J^t_{c_i,j}\right) + \overline{s}_{c_i} \tag{6.29}$$

$$\frac{\partial}{\partial t}\left(\overline{\rho}\widetilde{U}_i\right) + \frac{\partial}{\partial x_j}\left[\overline{\rho}\widetilde{U}_i\left(\widetilde{U}_j - U_{g,j}\right)\right] = -\frac{\partial \overline{p}}{\partial x_i} + \overline{\rho}\widetilde{f}_i + \frac{\partial}{\partial x_j}\left[\left(\overline{\tau}_{ij} + \tau^t_{ij}\right)\right] \tag{6.30}$$

$$\begin{aligned}\frac{\partial}{\partial t}\left(\overline{\rho}\widetilde{H}\right) &+ \frac{\partial}{\partial x_j}\left[\overline{\rho}\widetilde{H}\left(\widetilde{U}_j - U_{g,j}\right)\right] = \frac{\partial \overline{p}}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{q}_j + q^t_j\right) - \frac{\partial}{\partial x_j}\sum_{k=1}^{N_{sp}} \widetilde{h}_k \overline{J}_{c_k,j} + \overline{\widehat{J}_{c_k,j} h''_k} \\ &+ \frac{\partial}{\partial x_j}\left[\widetilde{U}_i\left(\overline{\tau}_{ij} + \tau^t_{ij}\right) + \overline{\tau}_{ij}\overline{u''_i} + \left(\overline{\tau'_{ij}u'_i} - \overline{\rho}\widetilde{k''u''_j}\right) - \overline{p}U_{g,j}\right] + \overline{\rho}\left(\widetilde{f}_i\widetilde{U}_i + \widetilde{f''_i u''_i}\right) + \overline{q}_v\end{aligned} \tag{6.31}$$

The unknown quantities appear in the species, momentum and total enthalpy equations:

- □ Turbulent mass diffusion flux

$$J^t_{c_i,j} = -\overline{\rho c''_i u''_j} = -\overline{\rho}\widetilde{c''_i u''_j} \tag{6.32}$$

- □ Turbulent momentum flux – Reynolds stress tensor

$$\tau^t_{ij} = -\overline{\rho u''_i u''_j} = -\overline{\rho}\widetilde{u''_i u''_j} \tag{6.33}$$

- □ Turbulent heat flux

$$q^t_j = -\overline{\rho h'' u''_j} = -\overline{\rho}\widetilde{h'' u''_j} \tag{6.34}$$

Note that specific enthalpy $\widehat{h}$ is usually defined in terms of of specific heat $\overline{c}_p$ and temperature $\widehat{T}$ for both single–component and multi–component phase, i.e. $\widehat{h} = \overline{c}_p\widehat{T}$. Therefore, the turbulent heat flux can be expressed as:

$$q^t_j = -\overline{\rho \overline{c}_p T'' u''_j} = -\overline{\rho}\;\overline{c}_p\widetilde{T'' u''_j} \tag{6.35}$$

- □ Work of body forces, i.e. production or destruction of turbulent kinetic energy by body force, see Equation (6.40)

$$P_b = \overline{\rho}\widetilde{f''_i u''_i} = \overline{\rho f''_i u''_i} \tag{6.36}$$

With regards to the corresponding molecular fluxes: $\overline{J}_{c_i,j}$, $\overline{\tau}_{ij}$ and $\overline{q}_j$, their constitutive relations are given by Fick's, Stokes's and Fourier's laws, respectively. The effects of property fluctuations appearing in the above constitutive laws (mass diffusivity $\mathscr{D}'_i$, viscosity μ' and thermal conductivity λ', respectively) are insignificant (cf. . Barre et al. [2002]; Huang et al. [1995]; Speziale [1996]). Further, the contributions of averaged fluctuating mass fractions $\overline{c''_i} = \overline{c}_i - \widetilde{c}_i$, velocities $\overline{u''_i} = \overline{U}_i - \widetilde{U}_i$ and temperatures $\overline{T''} = \overline{T} - \widetilde{T}$ should be negligible in regions where molecular transport is important. This means that the Favre averaged variables: $\widetilde{c}_i$, $\widetilde{U}_i$ and $\widetilde{T}$, which govern the constitutive relations, can be used.

The total enthalpy $\widetilde{H}$ in Equation (6.31) is defined as:

$$\widetilde{H} = \widetilde{E} + \frac{\overline{p}}{\overline{\rho}} = \widetilde{e} + \frac{\overline{p}}{\overline{\rho}} + \frac{\widetilde{U}_i\widetilde{U}_i}{2} + \widetilde{k} = \widetilde{h} + \frac{\widetilde{U}_i\widetilde{U}_i}{2} + \widetilde{k} \tag{6.37}$$

where the Favre averaged turbulent kinetic energy is defined as

$$\widetilde{k} = \frac{\overline{\rho u''_i u''_i}}{2\overline{\rho}} = \frac{\widetilde{u''_i u''_i}}{2} \tag{6.38}$$

It is useful to relate the definition of $\widetilde{k}$ with the Favre–fluctuating turbulent kinetic energy $k'' = u''_i u''_i/2 - \widetilde{k}$ and notice that $\overline{\rho k''} = 0$. Similarly, the Reynolds–averaged and Reynolds–fluctuating turbulent kinetic energy are defined as $\overline{k} = \overline{u'_i u'_i}/2$ and $k' = u'_i u'_i/2 - \overline{k}$, respectively. It can be shown that $\overline{k} = \widetilde{k} + \overline{k''} - \overline{u''_i}\;\overline{u''_i}/2$. In relation to the turbulent kinetic energy, Equation (6.40), the two terms in Equation (6.31), $\left(\overline{\tau'_{ij}u'_i} - \overline{\rho}\widetilde{k''u''_j}\right)$, correspond to viscous (molecular) diffusion and turbulent transport of $\widetilde{k}$, respectively.

One can derive the turbulent kinetic energy equation by averaging the instantaneous momentum Equation (6.8) multiplied by u''_i, see for example Wilcox [1998]:

$$\begin{aligned}\frac{\partial \overline{\rho}\widetilde{k}}{\partial t} + \frac{\partial}{\partial x_j}\left[\overline{\rho}\widetilde{k}\left(\widetilde{U}_j - U_{g,j}\right)\right] = {} & \underbrace{\tau^t_{ij}\frac{\partial \widetilde{U}_i}{\partial x_j}}_{P_k} - \underbrace{\overline{\tau'_{ij}\frac{\partial u'_i}{\partial x_j}}}_{\overline{\rho}\varepsilon} + \underbrace{\frac{\partial}{\partial x_j}\left(\overline{\tau'_{ij}u'_i} - \overline{\rho}\widetilde{k''u''_j} - \overline{p'u'_j}\right)}_{D_k} \\ & + \underbrace{\left(\overline{u''_i}\frac{\partial \overline{\tau}_{ij}}{\partial x_j} - \overline{u''_j}\frac{\partial \overline{p}}{\partial x_j} + \overline{p'\frac{\partial u'_j}{\partial x_j}}\right)}_{Compressibility\ terms} + \underbrace{\overline{\rho}\widetilde{f''_i u''_i}}_{P_b}\end{aligned} \tag{6.39}$$

The terms on the right–hand side of the above equation can be identified as:

- □ P_k: turbulent energy production
- □ $\overline{\rho}\varepsilon$: dissipation rate of $\widetilde{k}$
- □ D_k: contains viscous diffusion, turbulent transport and diffusion associated with pressure–velocity interactions
- □ Compressibility part includes the viscous stress and pressure work terms as well as the pressure dilatation term. These terms vanish in case of incompressible flow.

- P_b: turbulent energy production due to body forces

In order to close the compressible RANS equations *models* are required for:

- Turbulent mass diffusion flux $J^t_{c_i,j}$, Reynolds stress tensor τ^t_{ij} and turbulent heat flux q^t_j
- Turbulent mass flux, $\overline{u''_i}$
- Turbulent dissipation rate, ε
- Diffusion (D_k) and the pressure dilatation term $\overline{p' \partial u'_j / \partial x_j}$ in the turbulent kinetic energy, Equation (6.40)
- Production of turbulent kinetic energy by body force, P_b
- Additional terms in the total enthalpy equation related to turbulent transport of species, $\overline{\widehat{J}_{c_k,j} h''_k}$

The turbulent mass flux $\overline{u''_i}$ and associated compressibility terms in the turbulent energy equation, including the pressure dilatation correlation $\overline{p' \partial u'_j / \partial x_j}$, do not appear in the incompressible RANS equations. Modelling of RANS equations is dealt with in the turbulence modelling part of the manual.

In the following sections and generally throughout this manual, the Reynolds (ensemble) and Favre averaging operators, $\overline{(\ldots)}$ and $\widetilde{(\ldots)}$, respectively, will be used only for averages of fluctuating correlations. All flow variables without operators will be considered as averaged (mean–flow) quantities or instantaneous quantities in case of laminar flow where there is no turbulence.

7 MODELLING SPATIAL DOMAINS

The solver design enables simulations of momentum, mass and energy transport in multiple *fluid or solid domains*, which are parts of the single *computational domain*. In order to achieve this, the solver has a hierarchical domain structure, comprising:

□ material and fluid/solid domains,

□ domain boundaries and

□ domain interfaces.

The domain boundaries and interfaces are further decomposed into:

□ boundary regions and

□ interface regions.

These regions are used to implement physical boundary and interface conditions. Note that the word 'domain is often used to refer to any type of domain.

7.1 Multi–Domain Approach

The domain hierarchy is outlined below:

Computational Domain	⇒	This is the top-level domain which can be composed of any number of fluid and/or solid domains. Depending on the solution of energy equation, contiguous fluid and solid domains are grouped into a global domain. Currently, the computational and global domain is the same entity.
Global Domain	⇒	The global domain represents the solution domain for the energy equation if this equation needs to be solved *implicitly* over more than one fluid or solid domain.
Fluid / Solid Domain	⇒	The fluid domain is the same entity as fluid material domain, while the solid domain can consist of one or more solid material domains. The fluid domains must be separated by solid domains and vice versa.
Material Domain	⇒	The material domain is the 3D region of space filled in with the same continuum (general fluid or solid material). Note that the general fluid material can be either single–phase or multi–phase fluid, and each fluid phase can have more components (species).
Fluid Sub-Domain	⇒	This type is a part of the fluid domain. It can be used to activate certain features such as modelling of porous media, heat exchangers or fans. Modelling of rotational effects is not yet supported.

Mathematically, domains are defined by the corresponding volume meshes, while the domain boundaries and interfaces are described by the *boundary surface meshes* and the *interface meshes*, respectively. Thus the domain boundary is a collection of *boundary faces* and the interface is a collection of *internal faces* which are shared by two cells belonging to different materials. Prior to meshing, the CAD geometry defines the domain boundaries and interfaces. The domain boundary can not be shared between two material domains. In general, the actual *solution domain* for the given transport equation consists of one or more material domains.

As the solver operates on the non-overlapping cells it is important that the domain and sub–domain interfaces are represented by the union of internal cell faces which are defined by *identical* vertices, common to the neighbouring domain/sub–domain cells. Such domain interfaces and associated meshes will be referred as *conformal*. The non–conformal meshes can be converted into conformal ones by using arbitrary grid interface tool.

Simple examples of multi–domain simulations are separated fluid streams and conjugate heat transfer, where temperatures at the fluid–solid interfaces have to be coupled. Figure 7.1 can help to understand better VECTIS-MAX multi–domain structure. The material domain is the basic constituent – in this case there are two fluid materials, namely hot liquid (material 1) and cold air (material 2). Corresponding fluid domains coincide with these fluid materials. The fluid domains are separated by one solid domain, which in turn consists of two solid material domains (material 3 and 4). Thus, the whole computational domain has four material domains, grouped into two fluid domains and one solid domain. The first fluid domain (hot liquid) has three boundary regions: wall (at the left), inlet (at the top) and outlet (at the bottom). Similarly, the cold air fluid domain has three boundary regions: wall (at the right), inlet (at the bottom) and outlet (at the top). Finally, for each solid material domain there are two wall boundary regions (at the top and bottom). Two fluid/solid interface regions and one solid/solid region can be identified. If the energy equation is solved for all fluid and solid domains there will be one global domain, which coincides with the

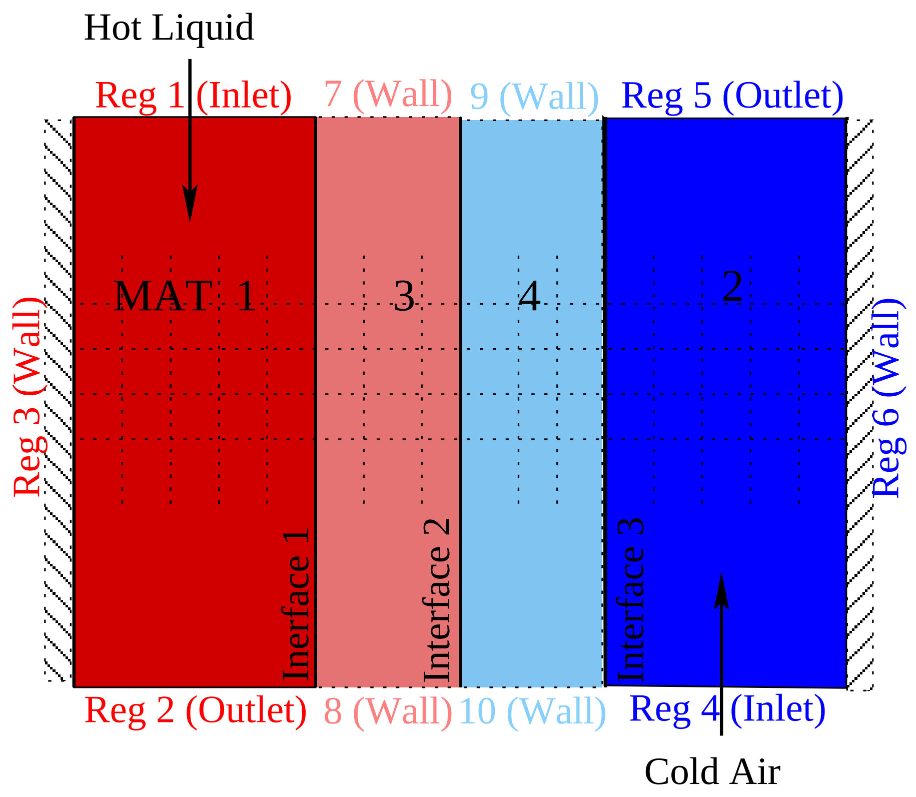


Figure 7.1: Simple multi–domain simulation example

computational domain.

7.2 Boundary Regions

Different physical boundary conditions are distinguished by Boundary Condition Types and they are applicable to different parts of the domain boundary. Therefore material domain boundaries are sub-divided into non–overlapping *boundary regions* according to the boundary condition type which will be applied later. A boundary region is described by a set of not necessarily adjacent boundary faces. Boundary regions are created directly on the domain geometry before the mesh generation, see Geometry In the future, it will be possible to create and manipulate boundary regions after the mesh generation.

7.3 Interface Regions

Interface Regions are implicitly defined at fluid/solid and solid/solid material interfaces. For every pair of adjacent material domains there is one interface region. With reference to the solution of transport equations the interface region can be treated as *explicit or implicit*. In case of the solution of energy equation over coupled domains (conjugate heat transfer) the corresponding interface regions are defined as implicit. For other equations they are explicit, i.e. they act as the standard wall boundary regions.

7.4 Ordering of Domain Components

VECTIS-MAX domain structure is built–up by indexing (labelling) its components with consecutive numbers. For every component (global domain, fluid/solid domain, material domain, boundary and interface region) the corresponding ordered set is created in the following way:

Global Domain Global domain is labelled with the index 1.

Fluid / Solid Domains Domain indices are first assigned to the fluid domains and then indexing continues with solid domains.

Material Domains Fluid material domains are ordered first, in the same way as the fluid domains. Then, solid material domains are indexed in the order which follows appearance of solid materials in the ordered set of solid domains.

Sub–Domains Indexing of sub–domains is done in the order as they appear in the ordered set of material domains.

Boundary Regions Indexing of boundary regions is done in the order as they appear in the ordered set of material domains.

Interface Regions Ordering of interface regions is based on the convention that an interface region is assigned to the material domain which has lower index than its neighbour. Thus, indexing is performed in the order which follows appearance of interface regions as one loops through the list of material domains.

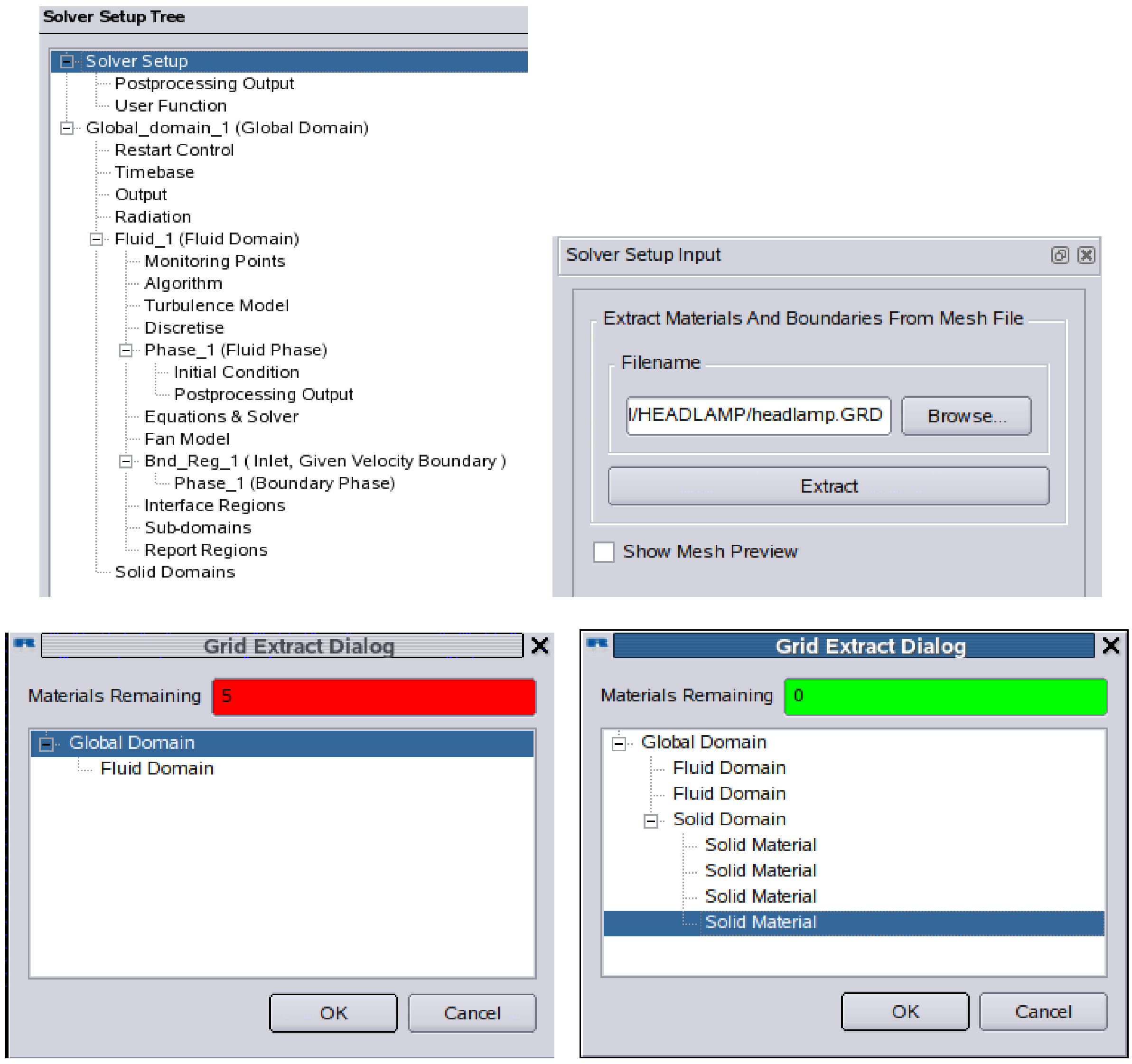
An example of ordered domain components is shown in Figure 7.1.

7.5 Creating a Domain Structure

Domain components are created through the process of mesh importation. The non–partitioned mesh produced by the pre–processing module *vpre* needs to be used. Such a mesh will have to

- □ describe the whole computational domain, consisting of all spatial material domains and their sub–domains,
- □ be re–ordered through all material domains and be conformal across material and sub–domain interfaces,
- □ and contain information about boundary regions and material sub–domains.

When a new project is started in *R–Desk* , the Solver Setup Tree, shown in Figure 7.2, will contain by default the one fluid domain (material) and one boundary region. The Solver Setup Input panel is then used to select the relevant grid file by using the Browse button. Once a file has been selected its name will appear in the Filename box. Clicking on the Extract button the Grid Extract Dialog pops up and displays a number of materials (material domains) found in the imported grid file. Note that these materials have already been ordered as explained in the section about ordering of domain



```
Solver Setup
    Postprocessing Output
    User Function
Global_domain_1 (Global Domain)
    Restart Control
    Timebase
    Output
    Radiation
    Fluid_1 (Fluid Domain)
        Monitoring Points
        Algorithm
        Turbulence Model
        Discretise
        Phase_1 (Fluid Phase)
            Initial Condition
            Postprocessing Output
        Equations & Solver
        Fan Model
        Bnd_Reg_1 ( Inlet, Given Velocity Boundary )
            Phase_1 (Boundary Phase)
        Interface Regions
        Sub-domains
        Report Regions
    Solid Domains
```

Filename: l/HEADLAMP/headlamp.GRD

Materials Remaining 5

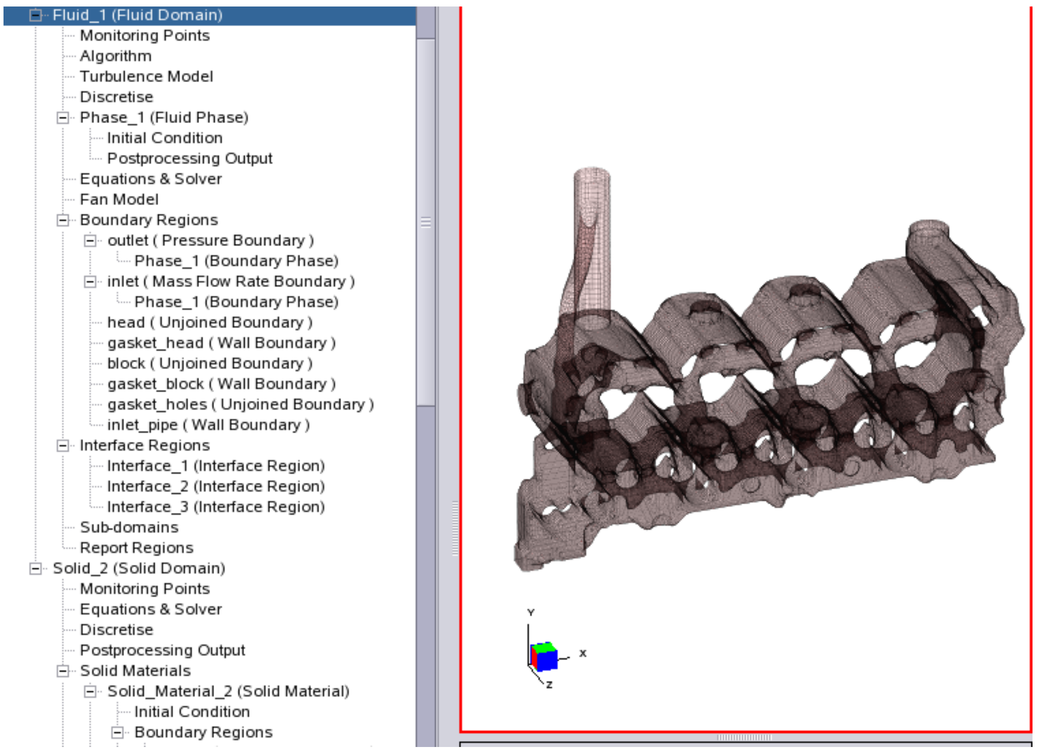
```
Global Domain
    Fluid Domain
```

Materials Remaining 0

```
Global Domain
    Fluid Domain
    Fluid Domain
    Solid Domain
        Solid Material
        Solid Material
        Solid Material
        Solid Material
```

Figure 7.2: *R–Desk* setup. Creating domain structure.

components. Respecting this ordering, corresponding fluid and solid domains need to be defined by allocating each fluid material from the grid file into a fluid domain and each solid material into a solid domain. By right clicking on the Global Domain entry additional fluid and solid domains can be created. Additional materials can be added to a solid domain by right clicking on the relevant Solid Domain . The number of remaining materials that have not been allocated is indicated in the Materials Remaining box. For instance, if two materials exist in the grid file but only one fluid domain has been created then there will be one material remaining. On the other hand, if more solid materials or fluids are defined than exist in the grid file then the number will be negative. When all materials have been assigned to the fluid and solid domains, the required computational domain components will be created by clicking on the OK button. The Solver Setup Tree will be updated showing new fluid/solid domains and materials as well as associated sub–domains and boundary and interface regions as Figure 7.3 and Figure 7.4 illustrate. At this stage, the default names have been assigned to the fluid/solid domains, sub–domains, fluid phases, solid materials and to all boundary and interface regions. If the relevant domain component from the tree (fluid

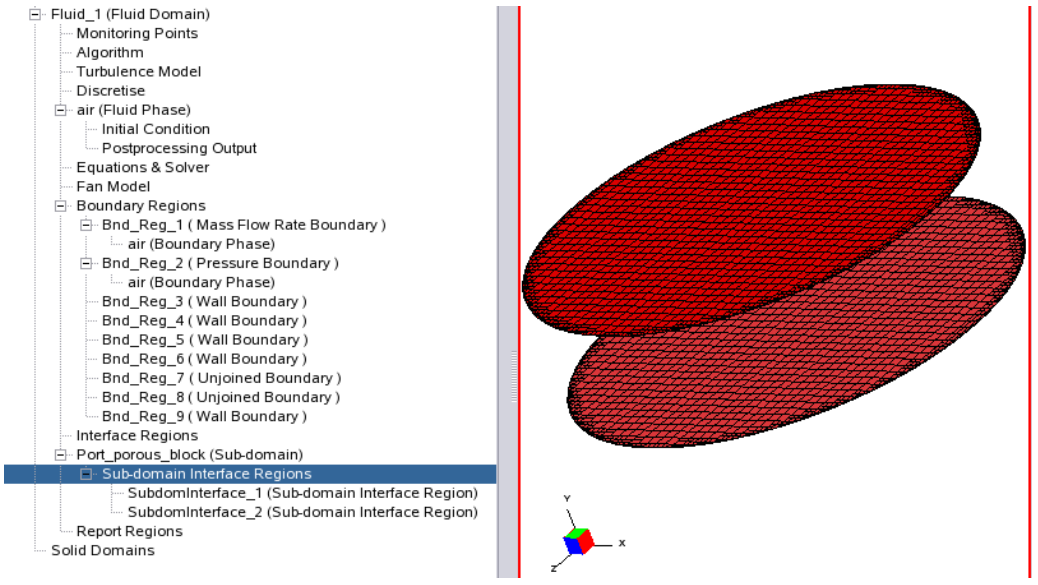


Fluid_1 (Fluid Domain)
- Monitoring Points
- Algorithm
- Turbulence Model
- Discretise
- Phase_1 (Fluid Phase)
  - Initial Condition
  - Postprocessing Output
- Equations & Solver
- Fan Model
- Boundary Regions
  - outlet (Pressure Boundary)
    - Phase_1 (Boundary Phase)
  - inlet (Mass Flow Rate Boundary)
    - Phase_1 (Boundary Phase)
  - head (Unjoined Boundary)
  - gasket_head (Wall Boundary)
  - block (Unjoined Boundary)
  - gasket_block (Wall Boundary)
  - gasket_holes (Unjoined Boundary)
  - inlet_pipe (Wall Boundary)
- Interface Regions
  - Interface_1 (Interface Region)
  - Interface_2 (Interface Region)
  - Interface_3 (Interface Region)
- Sub-domains
- Report Regions

Solid_2 (Solid Domain)
- Monitoring Points
- Equations & Solver
- Discretise
- Postprocessing Output
- Solid Materials
  - Solid_Material_2 (Solid Material)
    - Initial Condition
    - Boundary Regions

Figure 7.3: *R–Desk* setup. Example of created multi–domain structure (left) and previewed fluid domain (right) for conjugate heat transfer.

or solid domain, sub–domain, boundary or interface region or sub-domain interface) is selected it will be highlighted and shown in the preview window. In Figure 7.3, the fluid domain has been selected and previewed while Figure 7.4 previews a pair of sub–domain interface regions.



Fluid_1 (Fluid Domain)
- Monitoring Points
- Algorithm
- Turbulence Model
- Discretise
- air (Fluid Phase)
  - Initial Condition
  - Postprocessing Output
- Equations & Solver
- Fan Model
- Boundary Regions
  - Bnd_Reg_1 (Mass Flow Rate Boundary)
    - air (Boundary Phase)
  - Bnd_Reg_2 (Pressure Boundary)
    - air (Boundary Phase)
  - Bnd_Reg_3 (Wall Boundary)
  - Bnd_Reg_4 (Wall Boundary)
  - Bnd_Reg_5 (Wall Boundary)
  - Bnd_Reg_6 (Wall Boundary)
  - Bnd_Reg_7 (Unjoined Boundary)
  - Bnd_Reg_8 (Unjoined Boundary)
  - Bnd_Reg_9 (Wall Boundary)
- Interface Regions
- Port_porous_block (Sub-domain)
  - Sub-domain Interface Regions
    - SubdomInterface_1 (Sub-domain Interface Region)
    - SubdomInterface_2 (Sub-domain Interface Region)
- Report Regions

Solid Domains

Figure 7.4: *R–Desk* setup. Example of a single fluid domain tree with a porous sub–domain (left) and its sub–domain interface regions (right).

Note that in the case of multi–domain structure some boundary regions can have the "Unjoined Boundary" attribute. This attribute indicates that a small number of boundary faces of two adjacent boundary regions are non–conformal, i.e. they are not successfully converted into an interface region, see Arbitrary Grid Interface. In case of conjugate heat transfer the solver will treat these non-conformal wall boundaries as adiabatic walls.

8 MODELLING CONTINUA AND THEIR PROPERTIES

Physical properties of fluid or solid materials are very important constituents of the simulation. VECTIS-MAX defines the *general continuum* model which supports and manages *multiphase*, *multicomponent* fluid materials and physical models associated with them. The general continuum describes the mixture of phases, with each phase consisting of one or more species. It is important to understand the difference between the multicomponent phase (which is a mixture of species) and multiphase mixture. The principal assumption behind the definition of a multicomponent phase is that species are mixed at the molecular level. In contrast to the multicomponent phase, the multiphase mixture contains the single or multicomponent phases which are mixed at the macroscopic level, meaning that the corresponding length scales are much larger than the molecular ones.

Most of physical properties appearing in the transport equations are thermodynamic *intensive* properties. They arise from constitutive relations and from the equation of state, which are both required to close the system of transport equations.

8.1 Constitutive Relations & Transport Properties

Properties arising from the constitutive laws are often called transport properties. The constitutive laws define the viscous stress tensor τ_{ij}, heat flux vector q_i and mass flux vector of a chemical species J_i in terms of the velocity vector U_i, temperature T and species concentration (mass fraction) c, respectively.

8.1.1 Momentum transport in fluids and molecular viscosity

For *Newtonian fluids*, the constitutive relation between the viscous stresses τ_{ij} and the rates of deformation (strain tensor):

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right) \tag{8.1}$$

is given by the Stokes' law:

$$\tau_{ij} = 2\mu S_{ij} - \frac{2}{3}\mu\frac{\partial U_l}{\partial x_l}\ , \tag{8.2}$$

where μ denotes the molecular (dynamic) viscosity.

The viscosity describes the flow resistance of simple fluids. At moderate pressures, the viscosity in gases increases with increasing temperature whereas decreases in liquids. The *Sutherland* formula:

$$\mu = \mu_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + S_\mu}{T + S_\mu} , \tag{8.3}$$

based on a kinetic theory, is a good approximation for viscosity of dilute gases. The viscosity μ_0 is the viscosity at reference temperature T_0 while S_μ is an effective temperature in K (Sutherland constant). The following values are valid for air at moderate pressures and temperatures: $\mu_0 = 1.716 \times 10^{-5}\, kg/(ms)$, $T_0 = 273.15\,K$, and $S_\mu = 111\,K$. Another expression for dilute gases is the power–law formula:

$$\mu = \mu_0 \left(\frac{T}{T_0}\right)^{n} , \tag{8.4}$$

where μ_0 denotes the viscosity value at reference temperature T_0, and n is the exponent. For air at moderate pressures and temperatures: $\mu_0 = 1.716 \times 10^{-5}\, kg/(ms)$, $T_0 = 273.15\,K$, and $n = 2/3$.

The non—Newtonian fluids are not supported yet.

8.1.2 Energy transport and thermal conductivity

The heat conduction process is described by Fourier's law:

$$q_i = \lambda \frac{\partial T}{\partial x_i} , \tag{8.5}$$

which states that the heat flux vector is proportional to the temperature gradient. The constant of proportionality is the thermal conductivity λ. With increasing temperature the thermal conductivities of dilute (low density) gases increase and of most liquids decrease. The Sutherland formula, Equation (8.3), can be used for dilute gases:

$$\lambda = \lambda_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + S_\lambda}{T + S_\lambda} , \tag{8.6}$$

where S_λ is the effective temperature and λ_0 is the conductivity value at reference temperature $T_0 = 273.15\,K$.

Metals are generally better heat conductors than non-metals. The conductivity of most pure metals decreases with the temperature whereas the conductivity of non-metals increases. Some solid materials (fibrous and laminated solids) are characterised by the anisotropic thermal conductivity λ_{ij}, which is in general a tensor quantity (the heat flux vector is then given as $q_i = \lambda_{ij} \partial T/\partial x_j$). Materials with the anisotropic conductivity are not yet supported.

8.1.3 Mass transport and mass diffusion coefficients

Mass transport (diffusion) occurs in multicomponent flows where individual species move from a region of high concentration to a region of low concentration. The concentration or mass fraction of the i–th species c_i is defined as:

$$c_i = m_i/m\ , m = \sum_{i=1}^{N_{sp}} m_i\ , \sum_{i=1}^{N_{sp}} c_i = 1\ , \tag{8.7}$$

where m_i is the mass of i–th species , m is the total mass of the fluid phase occupying the control volume V, and N_{sp} is the number of species in the mixture. Since various species move at different velocities $\vec{U_{c_i}}$ the mass flux vector $\vec{J_{c_i}}$, relative to the local mass averaged velocity $\vec{U} = \sum_{i=1}^{N_{sp}} c_i \vec{U}_{c_i}$, is defined as:

$$\vec{J}_{c_i} = \rho\, c_i \left(\vec{U}_{c_i} - \vec{U} \right)\ , \tag{8.8}$$

where $\rho = m/V$ is the mass density of the mixture. In a laminar flow, the mass flux can be often modelled as:

$$\vec{J}_{c_i} = \rho\, \mathscr{D}_i \nabla c_i + \mathscr{D}_{T_i} \frac{\nabla T}{T}\ , \tag{8.9}$$

where $\mathscr{D}_i$ denotes the mass diffusion coefficient for i–th species and $\mathscr{D}_{T_i}$ is the thermal diffusion or Soret coefficient. The first term represents the *ordinary mass diffusion* described by Fick's first law of diffusion , whereas the second term is the *thermal diffusion*. In general, the ordinary mass diffusion is a complicated function of concentration gradients of all the mixture species and it is not always governed by Fick's law. Modelling such *fully multicomponent diffusion* is important for diffusion–dominated laminar flows (e.g. chemical vapour deposition) and it is not currently supported. The driving forces for the ordinary diffusion are concentration gradients, while the thermal diffusion is driven by a temperature gradient (examples are devices designed for separation of mixtures). There are additional measurable mass fluxes caused by a pressure gradient (*pressure diffusion* in centrifugal separators) and by external body forces (*force diffusion* in plasma technology). Normally, the ordinary diffusion term is dominant and other types of diffusion, including the thermal one, are currently neglected in VECTIS-MAX .

In turbulent flows, Equation (8.9) is replaced by:

$$\vec{J}_{c_i} = \left(\rho\, \mathscr{D}_i + \frac{\mu_t}{Sc_{i,t}} \right) \nabla c_i \tag{8.10}$$

where Sc_t is the turbulent Schmidt number:

$$Sc_{i,t} = \frac{\mu_t}{\rho\, \mathscr{D}_{i,t}}\ , \tag{8.11}$$

μ_t is the turbulent viscosity and $\mathscr{D}_{i,t}$ denotes the turbulent mass diffusion coefficient, see section about eddy viscosity formulation. The turbulent Schmidt number is of the order of unity (default value 0.9) and the same, constant values can be used for all species. As the turbulent diffusion usually overwhelms the laminar one, the constant values of laminar diffusion coefficients $\mathscr{D}_i$ are an appropriate choice.

8.2 Equation of State & Thermodynamic Properties

The thermodynamic state of a simple compressible system at equilibrium is fixed by any two independent thermodynamic properties. In CFD, pressure p and (absolute) temperature T are used as primary solution variables. Therefore, density ρ, internal energy e, thermal enthalpy $h = e + p/\rho$ and any other thermodynamic property ϕ could be determined as functions of p and T: $\rho = \rho(p,T)$, $e = e(p,T)$, $h = h(p,T)$ and $\phi = \phi(p,T)$. Note that transport properties discussed in the previous section are also true thermodynamic properties. With the help of exact differentials ($T\,ds$ equations), given as:

$$de = T\,ds - p\,dv = T\,ds + p\frac{d\rho}{\rho^2}\ ,\ \ dh = T\,ds + v\,dp = T\,ds + \frac{dp}{\rho} \tag{8.12}$$

(s is the specific entropy and $v = 1/\rho$ is the specific volume) various thermodynamic relations and properties can be defined.

8.2.1 Coefficients of expansion and compressibility

Considering the specific volume as a function $v = v(T,p)$, two thermodynamic properties related to the total differential dv are defined:

$$\beta = \frac{1}{v}\left(\frac{\partial v}{\partial T}\right)_p = -\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_p\ ,\ \mathscr{C} = -\frac{1}{v}\left(\frac{\partial v}{\partial p}\right)_T = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial p}\right)_T\ , \tag{8.13}$$

which are the coefficient of volumetric expansion β and the coefficient of (isothermal) compressibility $\mathscr{C}$. Considering an isentropic compression, the coefficient of isentropic compression:

$$\mathscr{C}_s = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial p}\right)_s \tag{8.14}$$

appears in the definition of a speed of sound a:

$$a = \sqrt{\left(\frac{\partial p}{\partial \rho}\right)_s} = \sqrt{\frac{1}{\rho\mathscr{C}_s}}\ , \tag{8.15}$$

8.2.2 Specific heat

Under certain conditions the specific heat is the amount of energy added by heat transfer to 1 kg of material to increase the temperature by 1 K. Depending on the process in which the energy is added, there is a distinction between the specific heat at *constant pressure*

$$c_p = \left(\frac{\partial h}{\partial T}\right)_p \tag{8.16}$$

and the specific heat at *constant volume*:

$$c_v = \left(\frac{\partial e}{\partial T}\right)_v \tag{8.17}$$

Their ratio:

$$\kappa = \frac{c_p}{c_v} \tag{8.18}$$

is called the specific heat ratio.

Employing the equation of state $p = p(T,\rho)$, i.e. $dp = (\partial p/\partial T)_\rho dT + (\partial p/\partial \rho)_T d\rho$, and $dh = de + d(p/\rho)$ one can get:

$$c_p - c_v = T\frac{\beta^2}{\rho\mathscr{C}} \text{ and } \left(\frac{\partial p}{\partial T}\right)_\rho = \frac{\beta}{\mathscr{C}} \tag{8.19}$$

8.2.3 Specific thermal energy and enthalpy

With the help of Equations (8.12) and considering first $e = e(T,v)$ and $s = s(T,v)$, and then $h = h(T,p)$, $s = s(T,p)$, the following thermodynamic relationships for the differential internal energy and enthalpy are obtained, respectively:

$$de = c_v dT + \left[p - T\left(\frac{\partial p}{\partial T}\right)_\rho\right]\frac{d\rho}{\rho^2} = c_v dT + \left(p - T\frac{\beta}{\mathscr{C}}\right)\frac{d\rho}{\rho^2} \ , \tag{8.20}$$

$$dh = c_p dT + \left[1 + T\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_p\right]\frac{dp}{\rho} = c_p dT + (1 - \beta T)\frac{dp}{\rho} \ . \tag{8.21}$$

Changes in specific internal energy and enthalpy between two states (1 and 2) are then given as:

$$e_2 - e_1 = \int_{T_1}^{T_2} c_v dT + \int_{\rho_1}^{\rho_2}\left(p - T\frac{\beta}{\mathscr{C}}\right)\frac{d\rho}{\rho^2} \tag{8.22}$$

$$h_2 - h_1 = \int_{T_1}^{T_2} c_p dT + \int_{p_1}^{p_2}(1 - \beta T)\frac{dp}{\rho} \ . \tag{8.23}$$

8.3 Thermally Perfect Fluids

Most of common gases and liquids can be considered *thermally perfect* in the sense that internal energy and thermal enthalpy depend on the temperature only. **Apart from phases declared to have WAVE properties, only thermally perfect fluids can be used in the current version.**

Selecting the state 1 as reference, with $e(T_{ref}) = h(T_{ref}) = 0$, the constitutive relations for internal energy and enthalpy (see Equations (8.22) and (8.23)) can be approximated as:

$$e(T) = \int_{T_{ref}}^{T} c_v dT = \overline{c_v}(T)\,T - \overline{c_v}(T_{ref})\,T_{ref} \ , \ \ \overline{c_v}(T) = \frac{1}{T}\int_0^T c_v(T)\,dT \tag{8.24}$$

$$h(T) = \int_{T_{ref}}^{T} c_p dT = \overline{c_p}(T)\,T - \overline{c_p}(T_{ref})\,T_{ref}, \ \ \overline{c_p}(T) = \frac{1}{T}\int_0^T c_p(T)\,dT \tag{8.25}$$

In the above equations $\overline{c_v}(T)$ and $\overline{c_p}(T)$ are temperature–averaged specific heats. Their use simplifies the calculation of internal energy and enthalpy of thermally perfect fluids. A general, polynomial function of temperature:

$$\phi(T) = a_1 + a_2 T + a_3 T^2 + ... + a_n T^{n-1} \ , \tag{8.26}$$

(n is the number of polynomial coefficients $a_i, i = 1, n$) is often useful to express specific heat (and other property) dependence on the temperature. Some fluid properties are based on the two commonly used thermally perfect fluid models: *incompressible substance model* and *ideal gas model*.

8.3.1 Incompressible substance model

For liquids and weakly compressible gases at moderate pressures, the density depends mainly on the temperature. The following expression, also known as Boussinesq density approximation, is appropriate for small temperature differences (for liquids temperature range less than $4K$, and for gases less than $15K$) :

$$\rho = \rho_{ref} - \rho_{ref}\beta\left(T - T_{ref}\right) , \tag{8.27}$$

where ρ_{ref} is the density at a reference temperature T_{ref}. The above expression is not valid for water around $4°C$. For liquids and solids it is often appropriate to assume constant density. Such an incompressible substance is characterised by $\beta = 0, \mathscr{C} = 0$, $c_p = c_v$, $\kappa = 1$, $de = c_v(T)dT$ or $e = \overline{c_v}T$ and $dh = c_p(T)dT + dp/\rho$ or $h = \overline{c_p}T + p/\rho$. The speed of sound becomes infinite since the coefficient of isentropic compressibility $\mathscr{C}_s$ tends to zero.

8.3.2 Ideal gas model

An ideal gas is described by the equation of state:

$$\rho = \frac{p}{RT}, \text{ where } R = R_u/M_w \tag{8.28}$$

is the gas constant, $R_u = 8314 J/(kmol\,K)$ is the universal gas constant and M_w denotes molecular weight. In case of incompressible and weakly compressible gas (Mach number $Ma < 0.3$), the reference pressure p_{ref} can be used instead of the local pressure p in the above equation:

$$\rho = \frac{p_{ref}}{RT}. \tag{8.29}$$

The following relationships hold for an ideal gas: $\beta = 1/T$, $\mathscr{C} = 1/p$, $\mathscr{C}_s = 1/(\kappa p)$, $c_p(T) - c_v(T) = \overline{c_p}(T) - \overline{c_v}(T) = R$, $\kappa(T) = c_p(T)/c_v(T)$, $de = c_v(T)dT$ or $e = \overline{c_v}T$, and $dh = c_p(T)dT$ or $h = \overline{c_p}T$. The speed of sound can be expressed as $a = \sqrt{\kappa RT}$ and c_p and c_v are related to κ and R as:

$$c_p(T) = \frac{\kappa}{\kappa - 1}R\ , \quad c_v(T) = \frac{1}{\kappa - 1}R\ . \tag{8.30}$$

For a number of common gases (Air, H_2, CO, O_2, H_2O, CO_2), c_p increases with temperature, while it is nearly constant ($2.5R$) for monatomic gases such as Ar, Ne and He. The polynomial form $(c_p/R) = f(T)$ involving five coefficients , Equation (8.26), is available in literature, Moran and Shapiro [1992]; Wark [1983]. It covers the temperature range from 300 to 1000 K and can be easily integrated to obtain the averaged specific heat $\overline{c_p}/R$ as a function of temperature. For air, these coefficients are: $a_1 = 3.653$, $a_2 = -1.337 \times 10^{-3}$, $a_3 = 3.294 \times 10^{-6}$, $a_4 = -1.913 \times 10^{-9}$, $a_5 = 0.2763 \times 10^{-12}$, giving $c_p(300K)/R = 3.499$ or $c_p = 1004.1\ J/(kgK)$, $\kappa = 1.400$.

Modelling of fully compressible or high–Mach flows ($Ma = U/a = U/\sqrt{\kappa RT} > 0.3$) often exploits the properties of ideal gases with *constant specific heats*. In this case, using Equations (8.12) and (8.30) one can describe the isentropic process between states 1 and 2 ($s_1 = s_2,\ \kappa = const$) by following relations:

$$\frac{T_2}{T_1} = \left(\frac{p_2}{p_1}\right)^{(\kappa-1)/\kappa} , \quad \frac{T_2}{T_1} = \left(\frac{\rho_2}{\rho_1}\right)^{\kappa-1} , \quad \text{or } \frac{p}{\rho^\kappa} = constant . \tag{8.31}$$

It is also convenient to work with properties evaluated at the *stagnation state*. The stagnation state is attained when a flowing fluid with $(U,\ h,\ T,\ p,\ \rho)$ decelerates to zero velocity isentropically. The corresponding properties are designated as *stagnation or total* properties. The total enthalpy and temperature are then defined as:

$$h_{tot} = h + \frac{U^2}{2} = e + \frac{p}{\rho} + \frac{U^2}{2} \tag{8.32}$$

$$T_{tot} = \frac{h_{tot}}{c_p} = T + \frac{U^2}{2c_p} \tag{8.33}$$

In case of an ideal gas, the total temperature and pressure are given, respectively:

$$T_{tot} = T\left(1 + \frac{\kappa-1}{2}Ma^2\right) \tag{8.34}$$

$$\frac{p_{tot}}{p} = \left(\frac{T_{tot}}{T}\right)^{\kappa/(\kappa-1)} , \quad p_{tot} = p\left(1 + \frac{\kappa-1}{2}Ma^2\right)^{\kappa/(\kappa-1)} . \tag{8.35}$$

For an incompressible fluid ($\rho = const$), the isentropic process toward the stagnation state results with

$$T_{tot} = T , \quad \text{and } p_{tot} = p + \rho\frac{U^2}{2} . \tag{8.36}$$

8.4 Properties of Multicomponent Phase

The physical properties of the multicomponent fluid phase (mixture of species) depend on the relative amounts of the components, usually specified in terms of mass fractions (concentrations), Equation (8.7). The **ideal mixtures** are supported in VECTIS-MAX . For such mixtures the species ("thermodynamic") density $\rho_i = m_i/V_i$ is related to the *partial volume of the species*, which is the volume V_i that would be occupied by mass m_i at the pressure and temperature of the mixture. As $\sum V_i = V$, the mixture density $\rho = m/V$ follows from the identity:

$$1 = \frac{\sum_{i=1}^{N_{sp}} V_i}{V} = \frac{\sum_{i=1}^{N_{sp}} m_i/\rho_i}{m/\rho} = \rho\sum_{i=1}^{N_{sp}}\frac{m_i}{m\rho_i} = \rho\sum_{i=1}^{N_{sp}}\frac{c_i}{\rho_i} , \tag{8.37}$$

and it is given as:

$$\rho = 1/\sum_{i=1}^{N_{sp}}\frac{c_i}{\rho_i} . \tag{8.38}$$

Other properties ϕ of the multicomponent phase are calculated from:

$$\phi = \sum_{i=1}^{N_{sp}} c_i \phi_i \, , \tag{8.39}$$

where ϕ_i is the property value of the i–th species. The above expression can be used to evaluate the composition–dependent laminar viscosity μ, specific heat c_p, thermal conductivity λ and gas constant of an ideal gas mixture R. Note that the specific volume of the mixture $1/\rho$ also conforms to Equation (8.39).

8.4.1 Reacting mixture flows

In a chemically reacting flow, where reactants disappear and products are formed, it is necessary to specify specific internal energy and enthalpy with reference to the *standard reference state*, usually defined at $T_{ref} = 298.15\,K$ and $p_{ref} = 101325\,Pa$. At this enthalpy datum, a value of zero is assigned to the *stable elements* (e.g. H_2, O_2, N_2) and the *enthalpy of formation* is assigned to *compounds* (e.g. water H_2O, carbon dioxide CO_2). The enthalpy of formation h^0, J/kg, is the energy released in an *exothermic reaction* ($h^0 < 0$) or absorbed in an *endothermic reaction* ($h^0 > 0$) when the compound is formed from its elements, all being at T_{ref} and p_{ref}. At a state (p,T) different than the standard state, the specific energy and enthalpy for thermally perfect fluids are defined as:

$$e_c = h^0 + e(p,T) - e(p_{ref},T_{ref}) = h^0 + \int_{T_{ref}}^{T} c_v \, dT = h^0 + \overline{c_v}(T)\,T - \overline{c_v}(T_{ref})\,T_{ref} \tag{8.40}$$

$$h_c = h^0 + h(p,T) - h(p_{ref},T_{ref}) = h^0 + \int_{T_{ref}}^{T} c_p \, dT = h^0 + \overline{c_p}(T)\,T - \overline{c_p}(T_{ref})\,T_{ref} \tag{8.41}$$

The above specific internal energy and enthalpy thus include the enthalpy of formation and they are often called thermo–chemical internal energy and enthalpy. The enthalpy of formation is required for each species i in order to define the mixture internal energy and enthalpy. Assuming that the standard state is the same for all species involved ($T_{ref,i} = T_{ref}$) we have:

$$e_c = \sum_i c_i e_{ci} = \sum_i c_i h_i^0 + \sum_i c_i \left[\overline{c_{v,i}} T - \overline{c_{v,i}} T_{ref,i}\right] = \sum_i c_i h_i^0 + \overline{c_v} T - \overline{c_v} T_{ref} \tag{8.42}$$

$$h_c = \sum_i c_i h_{ci} = \sum_i c_i h_i^0 + \sum_i c_i \left[\overline{c_{p,i}} T - \overline{c_{p,i}} T_{ref,i}\right] = \sum_i c_i h_i^0 + \overline{c_p} T - \overline{c_p} T_{ref} \, . \tag{8.43}$$

In the above equations, $\overline{c_v}$ and $\overline{c_p}$ denote the mixture specific heats, see Equation (8.39). For non–reacting mixtures, it is sufficient to work with thermal internal energy and enthalpy, in which case Equations (8.42) and (8.43) do not include enthalpy of formation terms and they reduce to Equations (8.24) and (8.25). Reacting mixture flows are not supported yet.

8.5 Properties of Multiphase Mixture

Multiphase flow modelling deals with flow situations where more than one fluid phase occupies the fluid domain. In comparison to the multicomponent fluid flow, the multiphase flow is mixed at the macroscopic level. The most common is the Euler–Euler modelling approach which requires the solution of *volume fraction* equations for fluid phases. For a small control volume V, the volume occupied by a fluid phase k, V_k, is $V_k = \alpha_k V$, where α_k is a volume fraction. The volume fraction is a function of space and time. As the volume of a phase can not be shared by another phase, the sum of all volume fractions is equal to one, $\sum_k \alpha_k = 1$. The mass fraction of the k–th phase, $c_k = m_k/m$, $m = \sum_k m_k$, is related to the volume fraction through the expression:

$$c_k = \frac{m_k}{m} = \frac{\rho_k V_k}{\rho_m V} = \alpha_k \frac{\rho_k}{\rho_m} \, , \tag{8.44}$$

where $\rho_k = m_k/V_k$ is the phase "thermodynamic" density, and ρ_m is the multiphase mixture density:

$$\rho_m = \frac{m}{V} = \sum_k m_k/V = \sum_k m_k \alpha_k/V_k = \sum_k \rho_k \alpha_k \tag{8.45}$$

Depending on the selected Euler–Euler model, each fluid phase can have its own flow field (the full Eulerian–Eulerian model) or it can share some common fields (VOF model for immiscible fluids and the mixture model). The properties that govern the transport equations of VOF and mixture models therefore depend on the properties of constituent fluid phases. In case of N_{ph} fluid phases, the volume–fraction mean properties ϕ_m are calculated in a similar way as the density, Equation (8.45):

$$\phi_m = \sum_{k=1}^{N_{ph}} \alpha_k \phi_k \, , \tag{8.46}$$

where ϕ_k denotes the property value of the constituent phase. Note that for the multicomponent phase the property ϕ_k should be calculated according to Equation (8.39).

8.6 Selecting Continuum and Its Properties

It follows from the previous sections that the physical properties can be associated with species, phases and with the mixture of phases. The association depends on the type of fluid flow:

Single–phase, single–component fluid. The phase is the same entity as the species (i.e. pure substance) and the phase properties appearing in the transport equations are required.

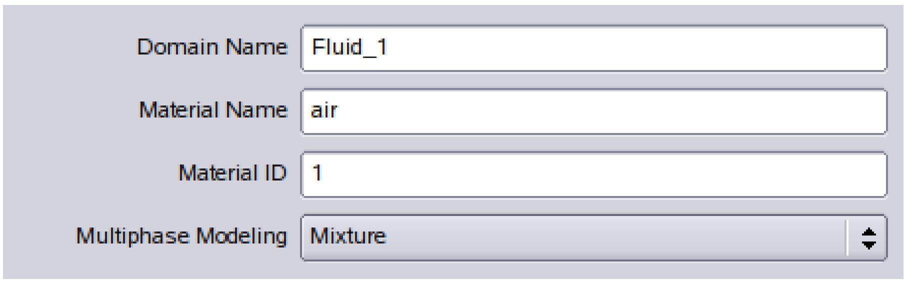
Single–phase, multicomponent fluid. Some species properties are required to solve mass fraction equations and others to calculate the properties of the multicomponent fluid phase, see Equation (8.39). The latter are needed for the fluid phase transport equations.

Multiphase, multicomponent fluid. The properties required depend on the multiphase flow modelling:

- Full Euler–Euler model: Phase properties are required. For the multicomponent phases, the properties can be derived from constituent species properties, Equation (8.39)
- Mixture and Volume of Fluid (VOF) model: Properties of the multiphase mixture must be defined, see Equation (8.46), from properties of constituent phases. For the multicomponent phases, the properties can be calculated from constituent species properties, Equation (8.39).

8.6.1 Setting a multiphase mixture

After importing the mesh and creating the domain structure, the single–phase, single-component fluid will be assigned to fluid domains. In order to define the multiphase mixture, left-click on a Fluid Domain node in the Solver Setup Tree . This will open the Fluid Domain Setup whose top part is shown in Figure 8.1. Here, the default Domain and Material names can be changed. The Material ID value is currently not used. Left–click on the Multiphase Modelling button and select Mixture .



Domain Name	Fluid_1
Material Name	air
Material ID	1
Multiphase Modeling	Mixture

Figure 8.1: *R–Desk* setup. Selecting a multiphase model.

The Solver Setup Tree will be refreshed and the user is now able to create a multiphase mixture by adding the required number of fluid phases. To add a phase, right–click on the Fluid Phase node and select Add Fluid Phases , Figure 8.2.

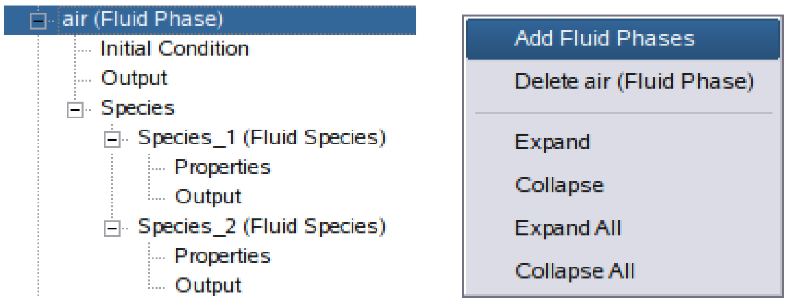
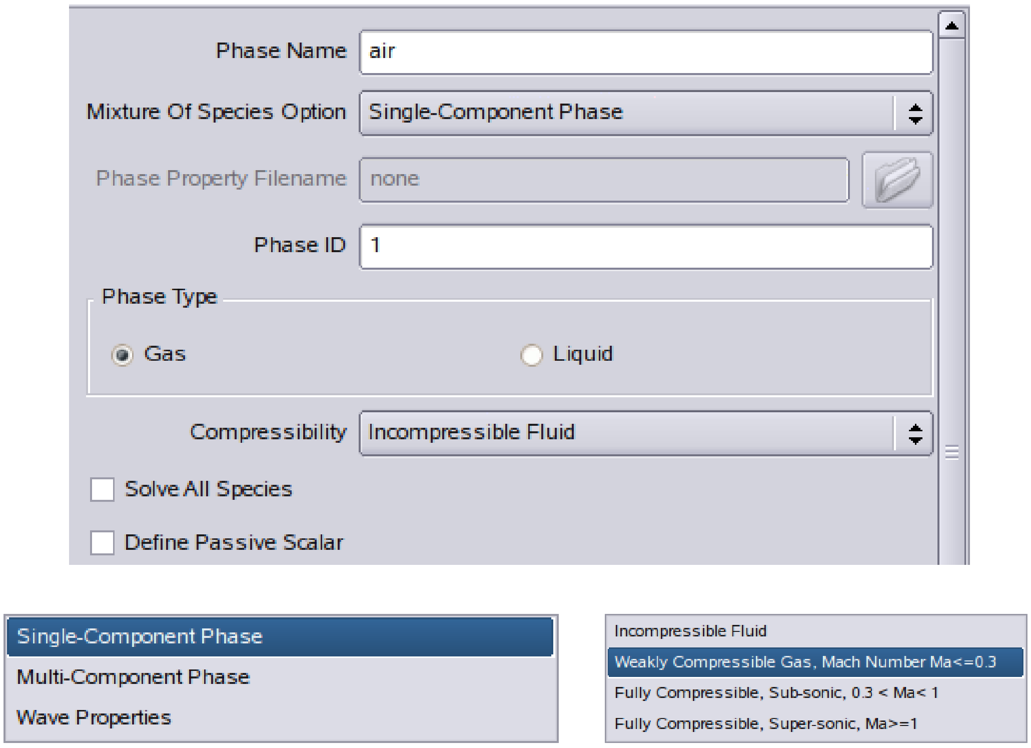


Figure 8.2: *R–Desk* setup. Adding a fluid phase.

8.6.2 Setting a phase and its properties

To define a phase model the Fluid Phase Setup panel needs to be open by left-click on the Fluid Phase node. The top Phase Name edit box can be used to enter the phase name. As Figure 8.3 (top) shows, the single–component phase is the default selection. Clicking on the Mixture of Species Option box, a panel pops up, Figure 8.3 (bottom, left), where three phase models can be selected: single–component phase, multi–component phase and a phase belonging to the WAVE data base. The Phase Type can be selected by using the Gas or Liquid radio button. If the Gas is selected then the phase compressibility should be defined by clicking on the Compressibility list



Phase Name: air
Mixture Of Species Option: Single-Component Phase
Phase Property Filename: none
Phase ID: 1
Phase Type: [✓ Gas] Liquid
Compressibility: Incompressible Fluid
☐ Solve All Species
☐ Define Passive Scalar

Single-Component Phase
Multi-Component Phase
Wave Properties
Incompressible Fluid
Weakly Compressible Gas, Mach Number Ma<=0.3
Fully Compressible, Sub-sonic, 0.3 < Ma< 1
Fully Compressible, Super-sonic, Ma>=1

Figure 8.3: *R–Desk* setup. Setting up a fluid phase.

box. The pop up panel, shown in Figure 8.3(bottom, right), will contain four options to describe the fluid phase in terms of compressibility (see Equation 8.13), i.e. in terms of Mach number:

Incompressible Fluid Density is constant, compressibility coefficient $\mathscr{C} \to 0$.

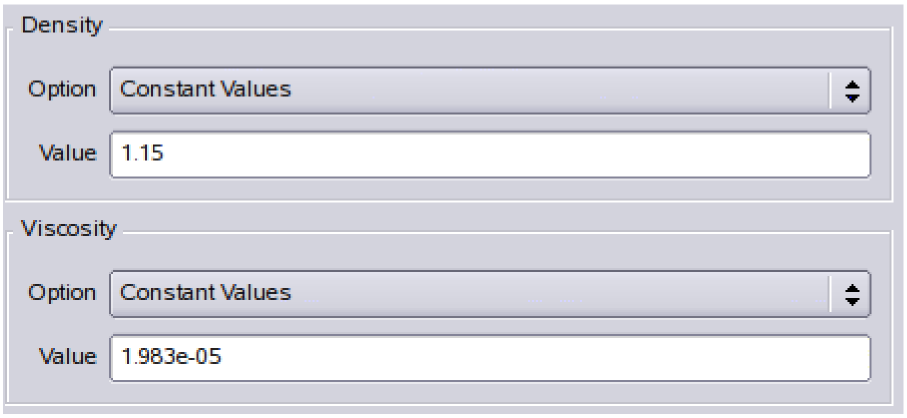
Weakly Compressible Gas Density may vary with temperature but the change of the compressibility coefficient is relatively small. Pressure changes relative to the reference pressure are small. In terms of the Mach number, weakly compressible fluid is broadly defined for $Ma < 0.3$.

Fully Compressible, Subsonic $0.3 \leq Ma \leq 1$.

Fully Compressible, Supersonic $Ma > 1$.

Single–component phase. The lower part of the Fluid Phase Setup panel provides several sub–panels to specify all required thermo–physical properties, namely Density , Viscosity , Conductivity , Specific Heat , Molecular Weight and Thermal Expansion Coefficient . The sub–panels for setting the density and viscosity are depicted in Figure 8.4(top). For each property there is the Option list box which displays calculation options available for the considered property. For example, left–click on the Viscosity Option will display the viscosity calculation options given in Figure 8.4 (bottom). Constant property values are the default options and these values are entered in the corresponding Value line box. For properties dependent on the temperature T, various functions $f(T)$ are available. Property calculation options are summarised in Table 8.1.

Piecewise functions. A property can be specified in terms of the following piecewise functions:



Density

Option: Constant Values

Value: 1.15

Viscosity

Option: Constant Values

Value: 1.983e-05

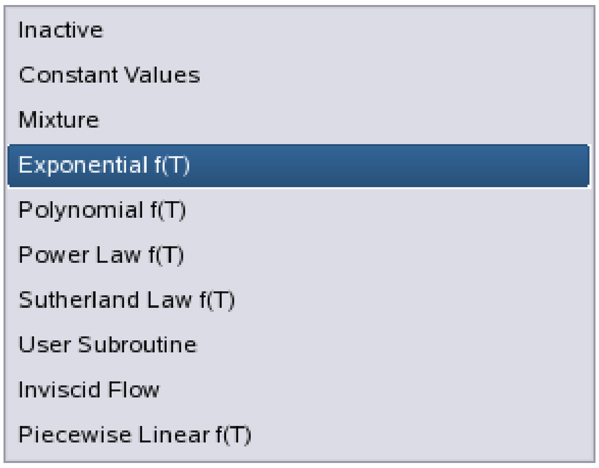


Figure 8.4: *R–Desk* setup. Setting properties of the single–component phase.

- piecewise linear functions and
- piecewise polynomial functions

Piecewise linear functions. A material property can be described using piecewise linear functions where each function represents a linear variation of a property ϕ and in general form is defined as:

$$\phi(T) = \phi_n + \frac{\phi_{n+1} - \phi_n}{T_{n+1} - T_n}(T - T_n) \tag{8.47}$$

Figure 8.5 represents a material property in terms of piecewise linear functions. The number of points in this example is 3 and the number of intervals is 2. If the temperature value T is in interval $[T_n, T_{n+1})$ then the piecewise linear function is that given in Equation 8.47. If the temperature value T is in interval $[T_{n+1}, T_{n+2})$ then Equation 8.47 takes the form:

$$\phi(T) = \phi_{n+1} + \frac{\phi_{n+2} - \phi_{n+1}}{T_{n+2} - T_{n+1}}(T - T_{n+1}) \tag{8.48}$$

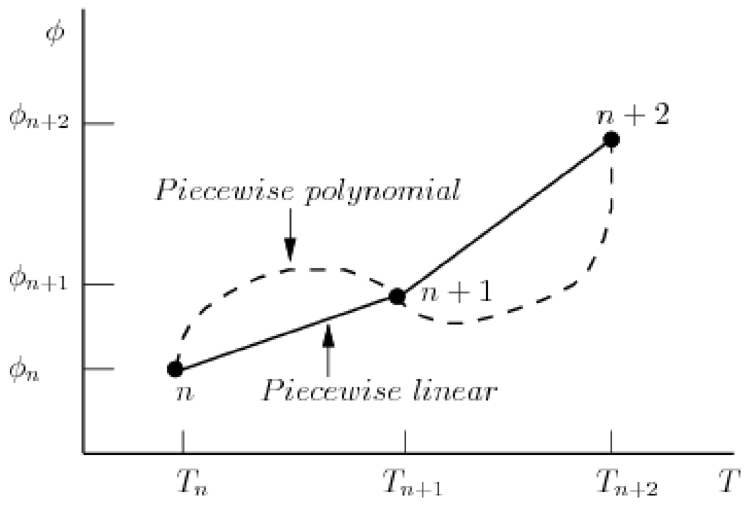
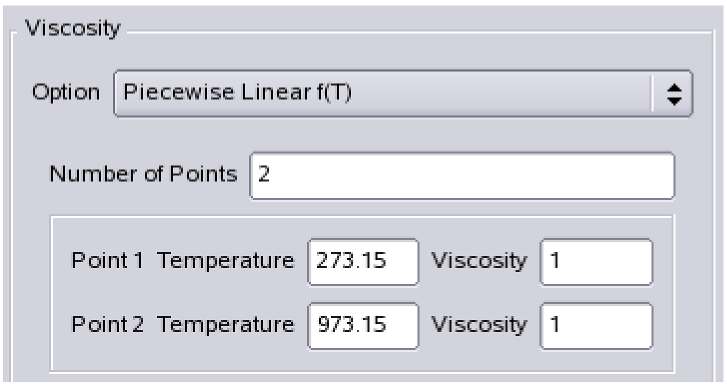


Figure 8.5: A material property in terms of piecewise linear & polynomial functions



Viscosity

Option: Piecewise Linear f(T)

Number of Points: 2

Point 1 Temperature	273.15	Viscosity	1
Point 2 Temperature	973.15	Viscosity	1

Figure 8.6: *R–Desk* setup. Property calculation options: Piecewise linear.

Figure 8.6 shows the piecewise linear option having the minimum Number of Points =2 . Temperature at Point 1 and Point 2 corresponds to T_n and T_{n+1} in relation 8.47 respectively. Viscosity at Point 1 and Point 2 corresponds to ϕ_n and ϕ_{n+1} in relation 8.47.

Piecewise polynomial functions. A material property can also be defined with a polynomial function of temperature and may consist of one or more polynomial functions defined in each range or interval. Consider 2 piecewise polynomial functions defined with intervals $[T_n, T_{n+1})$ and $[T_{n+1}, T_{n+2})$ as in Figure 8.5. In general form a polynomial function in interval $[T_n, T_{n+1})$ is defined as:

$$\phi(T) = a_1 + a_2 T + a_3 T^2 + a_4 T^3 + ... \tag{8.49}$$

with the following constraint: $T_n \leq T < T_{n+1}$.

Similarly for interval $[T_{n+1}, T_{n+2})$ the general form of the polynomial function is:

$$\phi(T) = b_1 + b_2 T + b_3 T^2 + b_4 T^3 + ... \tag{8.50}$$

with the following constraint: $T_{n+1} \leq T < T_{n+2}$.

Thus, the general form to specify a material property using any number of piecewise polynomial functions is given in relation 8.51.

$$\phi(T) = \begin{cases} a_1 + a_2T + a_3T^2 + a_4T^3 + \ldots & \text{for } T_n \le T < T_{n+1} \\ b_1 + b_2T + b_3T^2 + b_4T^3 + \ldots & \text{for } T_{n+1} \le T < T_{n+2} \\ c_1 + c_2T + c_3T^2 + c_4T^3 + \ldots & \text{for } T_{n+2} \le T < T_{n+3} \\ \vdots & \end{cases} \tag{8.51}$$

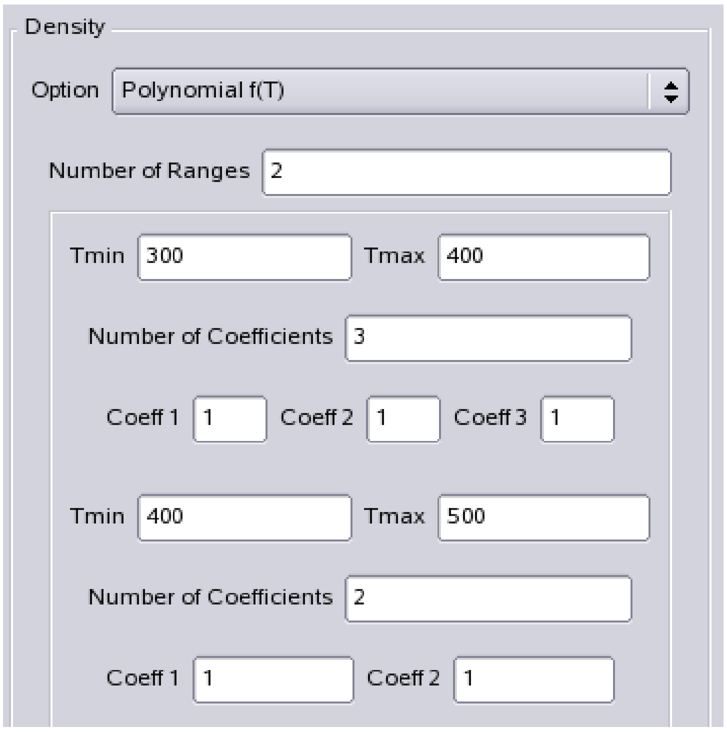

Density
Option: Polynomial f(T)
Number of Ranges: 2
Tmin: 300 Tmax: 400
Number of Coefficients: 3
Coeff 1: 1 Coeff 2: 1 Coeff 3: 1
Tmin: 400 Tmax: 500
Number of Coefficients: 2
Coeff 1: 1 Coeff 2: 1

Figure 8.7: *R–Desk* setup. Property calculation options: Polynomial/ Piecewise polynomial.

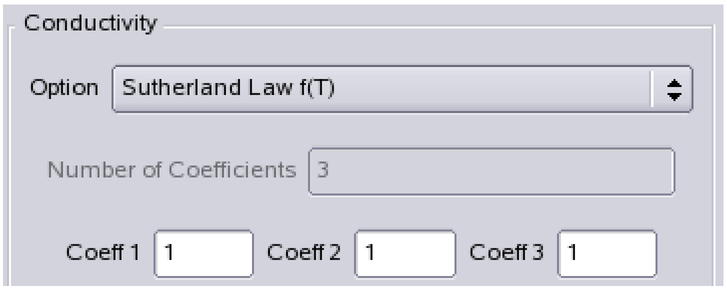
Figure 8.7 shows the polynomial option where the Number of Ranges defines how many piecewise polynomial functions would be used. The default is 1. For each range, the minimum and maximum temperature (Tmin and Tmax) should be specified continuously. Each range can contain any Number of Coefficients that can be inserted in the appropriate edit boxes. For example in Figure 8.7 for range 1, Coeff 1 , Coeff 2 and Coeff 3 correspond to a_1, a_2 and a_3 in relation 8.51 respectively whereas for range 2, Coeff 1 and Coeff 2 correspond to b_1 and b_2 in relation 8.51 respectively.

Other temperature functions $f(T)$ in Table 8.1 are defined in terms of the coefficients c_n. Such an example is the Sutherland formula. As Figure 8.8 shows, these coefficients should be specified in the available edit boxes: Coeff 1 corresponds to the c_1 in the Table 8.1, etc.

Multi–component phase. If the fluid phase has been defined as the multi–component phase (see Figure 8.3) the Solver Setup Tree will show Species node with two defined species, Figure 8.9 (left). Note that Mixture calculation option should be assigned by default to all phase properties. If the user specifies a different option than Mixture it means that the given phase property will not depend on the composition of the mixture, i.e. Equation (8.39) will not be used.

Option	Expression	No of Coeffs	Intended Usage
Constant values	-	-	All properties
Boussinesq	-	-	Density, see Eq.(8.27)
Exponential $f(T)$	$c_3 + c_4 \exp(c_1 + c_2 T)$	4	All properties
Inviscid	-	-	Viscosity
Piecewise Linear $f(T)$	As above		All properties
Polynomial $f(T)$	As above		All properties
Ideal Gas $f(T)$	$c_1 + c_2 T + \ldots + c_n T^{n-1}$	5, Wark [1983]	Specific heat for common gases; density: Eq. (8.28)
Power Law $f(T)$	$c_1 \left(\frac{T}{c_2}\right)^{c_3}$	3	Viscosity, Eq.(8.4)
Sutherland Law	$c_1 \left(\frac{T}{c_2}\right)^{3/2} \frac{c_2+c_3}{T+c_3}$	3	Viscosity, Conductivity: Eqs (8.3), (8.6)
Mixture	-	-	Multi–component phase
User Subroutine	-	-	All properties

Table 8.1: Options for calculation of single–component phase and species properties.


Conductivity

Option: Sutherland Law f(T)

Number of Coefficients: 3

Coeff 1: 1 Coeff 2: 1 Coeff 3: 1

Figure 8.8: *R–Desk* setup. Property calculation options: Sutherland formula.

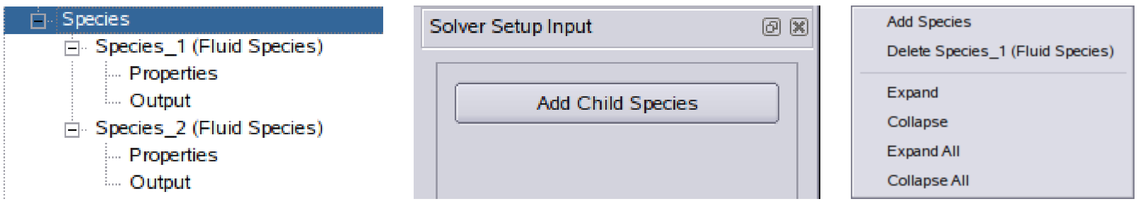
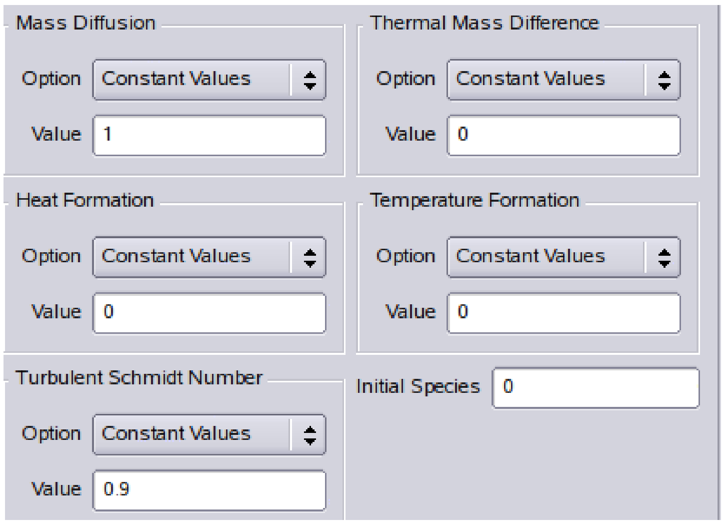


Figure 8.9: *R–Desk* setup. Defining the multi–component phase in terms of species.

Left–click on the Species node opens a new panel, Figure 8.9 (middle), where a new species can be added to the mixture by clicking on the Add Child Species push button. In this case, the species tree will be refreshed and show just added species. Right–click on any Fluid Species node pops up a panel, Figure 8.9(right), which can be used to add a new species or delete the current species if the mixture contains more than two species. Physical properties of each species are specified in the Species Properties panel which is displayed after clicking on the Properties node of the considered species, see Figure 8.9 (left). In addition to the properties of single–component phase, this panel, shown in Figure 8.10, contains sub–panels to specify calculation options for the Mass

Diffusion coefficient, Thermal Mass Diffusion , Heat Formation , Temperature Formation and Turbulent Schmidt Number . The heat (enthalpy) of formation and the temperature of formation are relevant to the reacting mixture flows.


Mass Diffusion
Option: Constant Values
Value: 1

Thermal Mass Difference
Option: Constant Values
Value: 0

Heat Formation
Option: Constant Values
Value: 0

Temperature Formation
Option: Constant Values
Value: 0

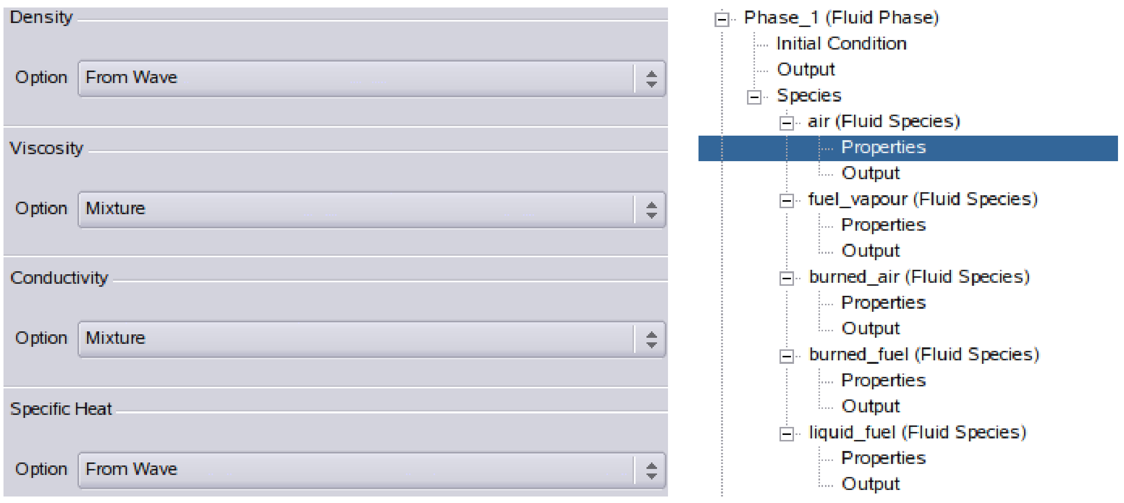
Turbulent Schmidt Number
Option: Constant Values
Value: 0.9

Initial Species: 0

Figure 8.10: *R–Desk* setup. Setting properties of individual species.

The calculation options for species properties are specified in a similar way as for the single–component fluid, in accordance with Table 8.1. The Initial Species edit box is provided to specify the initial value of species mass fraction.

Phase with WAVE properties. As Figure 8.3 indicates, a phase can have the WAVE attributes which means it is a multi–component fluid phase consisting of five species (components): air, fuel vapour, burned air, burned fuel and liquid fuel. When this type of phase is selected the Fluid Phase Setup panel, Figure 8.11 (left), will display the From Wave option for a Density and Specific Heat . For all other properties the Mixture option is pre–defined.


Density
Option: From Wave

Viscosity
Option: Mixture

Conductivity
Option: Mixture

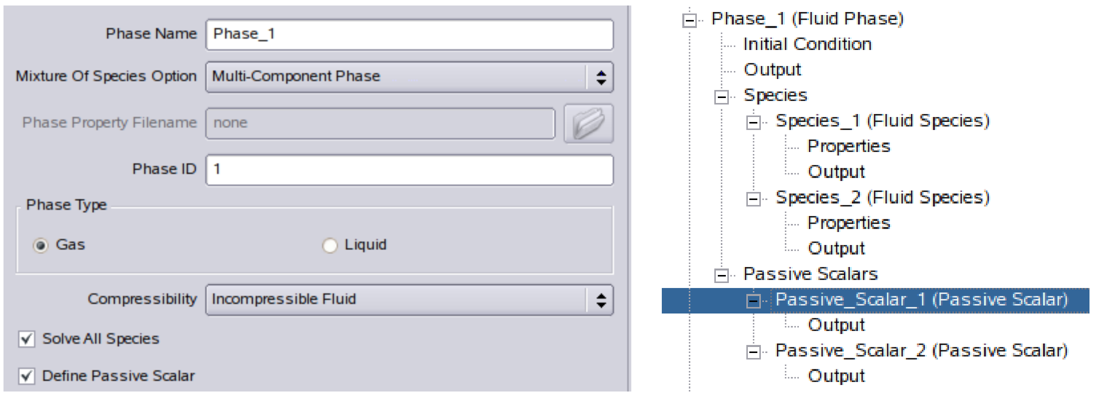
Specific Heat
Option: From Wave

Phase_1 (Fluid Phase)
- Initial Condition
- Output
- Species
  - air (Fluid Species)
    - Properties
    - Output
  - fuel_vapour (Fluid Species)
    - Properties
    - Output
  - burned_air (Fluid Species)
    - Properties
    - Output
  - burned_fuel (Fluid Species)
    - Properties
    - Output
  - liquid_fuel (Fluid Species)
    - Properties
    - Output

Figure 8.11: *R–Desk* setup. Setting properties of WAVE components (species).

The refreshed Solver Setup Tree will present the Species node with the above WAVE species, see Figure 8.11(right). Left–clicking on the Properties node of the selected Fluid Species opens the Species Properties panel similar to the one shown in Figures 8.4 and 8.10. Here, the phase properties labelled with the Mixture option can be specified in the same way as for any general species. Density and specific heat will show the Inactive option.

Passive scalar and its properties. Passive scalars are similar to species, however they do not affect properties of the parent phase. The Fluid Phase Setup panel contains the Define Passive Scalar check box as shown in Figure 8.12 (left).

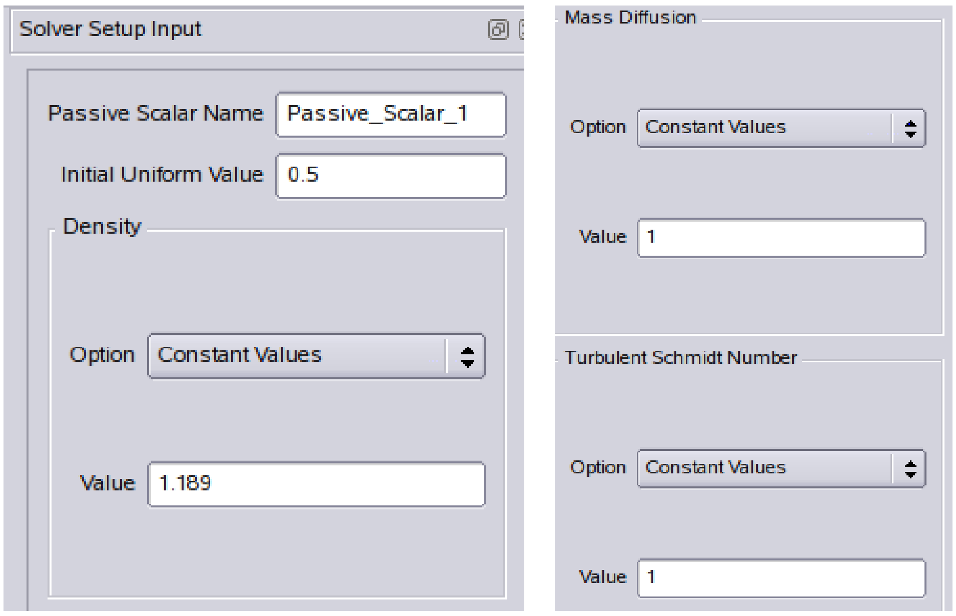


Phase Name	Phase_1
Mixture Of Species Option	Multi-Component Phase
Phase Property Filename	none
Phase ID	1
Phase Type	[✓ Gas] Liquid
Compressibility	Incompressible Fluid
✓ Solve All Species	
✓ Define Passive Scalar	

```
Phase_1 (Fluid Phase)
    Initial Condition
    Output
    Species
        Species_1 (Fluid Species)
            Properties
            Output
        Species_2 (Fluid Species)
            Properties
            Output
    Passive Scalars
        Passive_Scalar_1 (Passive Scalar)
            Output
        Passive_Scalar_2 (Passive Scalar)
            Output
```

Figure 8.12: *R–Desk* setup. Adding passive scalars.

Left–click on this box adds the Passive Scalars node to the Fluid Phase node in the Solver Setup Tree , Figure 8.12 (right). Only one passive scalar is added by this action. Additional ones can be added by right–click on the Passive Scalar node(s). For more than one passive scalar, the right–click will display a pop up panel with an option to delete the current scalar. To access property definition of passive scalars, the user should left–click on the Passive Scalar node. This action opens the Passive Scalar Properties panel, Figure 8.13.


Solver Setup Input

Passive Scalar Name: Passive_Scalar_1

Initial Uniform Value: 0.5

Density

Option: Constant Values

Value: 1.189

Mass Diffusion

Option: Constant Values

Value: 1

Turbulent Schmidt Number

Option: Constant Values

Value: 1

Figure 8.13: *R–Desk* setup. Setting properties of passive scalars.

In this panel, Passive Scalar Name can be edited and Initial Uniform Value specified. The required physical properties are Density , Mass Diffusion coefficient and Turbulent Schmidt Number . These properties should be defined in a similar way as for the standard species.

9

MODELLING TURBULENCE

9.1 Introduction

Turbulence is commonly found in all fluid flows in nature and engineering. It is “the chief outstanding difficulty of our subject” (cf. Lamb [1932] and “. . . the last unsolved problem of classical physics” (attributed to R. Feynman). Bradshaw [1994] prefers to define turbulence as the general solution of the Navier-Stokes equations: “...it is entirely true, and it adds nothing to what was known already,”i.e. that turbulent flows are three-dimensional and unsteady, irregular, rotational, diffusive and dissipative. Turbulence appears when the local viscous force can’t suppress random disturbances and flow instabilities amplified by the inertial, buoyancy and other external forces. In practice, the minimum (critical) values of non–dimensional numbers:

- ☐ Reynolds number $Re = \rho UL/\mu$,
- ☐ Rayleigh $Ra = \beta\Delta T g L^3 \rho^2 c_p/(\mu\lambda)$
- ☐ or Grashof number $Gr = Ra/Pr$ ($Pr = \mu c_p/\lambda$ – Prandtl number)

are used as criteria for turbulence generation. The above numbers are based on the characteristic flow variables: velocity U, dimension L and the temperature difference ΔT. For values of non–dimensional numbers below the critical ones, a laminar flow exists.

Direct Numerical Simulation (DNS) of Navier-Stokes equations is intractable for most turbulent flows as all length and time scales must be resolved numerically. The smallest of these scales are associated with *dissipative eddies* which are responsible for the dissipation of turbulence mechanical energy into heat due to viscosity. They are known as Kolmogorov length and time scales ℓ_k and T_k:

$$\ell_k = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4}, \; T_k = \left(\frac{\nu}{\varepsilon}\right)^{1/2}, \; \nu = \frac{\mu}{\rho} \; . \tag{9.1}$$

where ε represents the viscous dissipation rate per unit mass and ν is the kinematic viscosity. The large *energy containing eddies* (which are close to the largest eddies) extract energy from the mean–flow and start *turbulence cascade*, where energy is transferred towards smaller and smaller eddies until final dissipation by viscosity. The length and time scales of these large “energetic eddies” can be defined in terms of turbulence kinetic energy k and its dissipation rate ε:

$$T_t = \frac{k}{\varepsilon}, \; \ell_t = \frac{k^{3/2}}{\varepsilon} \tag{9.2}$$

The time and length scales are combined to define the characteristic Kolmogorov and turbulent velocity scales $v_k = (\nu\varepsilon)^{1/4}$ and $v_t = k^{1/2}$, respectively. In addition, the turbulence Reynolds number can be defined as

$$Re_t = \frac{k^{1/2}\ell_t}{\nu} = \frac{\rho k^2}{\mu} \tag{9.3}$$

Note that the turbulence Reynolds number based on the Kolmogorov scales is $Re_k = 1$.

>From the relation $\ell_t/\ell_k = Re_t^{3/4}$ it is clear that the range of scales increases with the turbulence Reynolds number or with the mean–flow Reynolds number Re as $Re \propto Re_t$ ($l_t \propto L$ and $k^{1/2} \propto U$). One can estimate a minimum number of computational cells as $N_{cell} \propto (\ell_t/\ell_k)^3 \propto Re^{9/4}$. At present, the DNS can be used for flows in simple geometries, with Re up to 10^4. Another route, which still requires too much computational efforts but the next best alternative to DNS, is *Large Eddy Simulation* (LES). The LES resolves the large-scale eddy motion in space and time while the sub–scale motion, defined in terms of numerical mesh, requires statistical modelling. Consequently, it can be applied to more complex geometries and higher Reynolds numbers. While DNS is valuable research tools, the solution of Reynolds Averaged Navier–Stokes (RANS) equations has been the only viable approach in industrial CFD for over three decades. LES might eventually emerge as the future industrial standard but it can be argued (cf. Hanjalic [2005]) that for long RANS will play an important role, especially in industrial applications. With appearance of some *hybrid RANS/LES* methods, which combine advantages of RANS and LES for wall–bounded flows, it is expected to see more extensive use of advanced RANS methods, Hanjalic [2005].

VECTIS-MAX employs the RANS approach and currently provides closure of the RANS equations via linear two–equation $k-\varepsilon$ models. The reminder of this section contains:

- An overview of turbulence models for RANS
- Description of linear two–equation $k-\varepsilon$ models implemented in VECTIS-MAX
- Near–wall turbulence treatment
- How to select a model and wall treatment

9.2 Overview of Turbulence Models for RANS

It has been shown that RANS equations contain unknown quantities, statistically *one point second moments* $\overline{\rho}\widetilde{\varphi'' u_j''}$, representing turbulent mass diffusion, momentum and heat flux for φ equal to c_i, u_i and T, respectively. Consequently, the *turbulence closure problem* arises, that is these quantities have to be determined by a turbulence model. The current turbulence models can be basically classified into two groups, namely:

Eddy Viscosity/Diffusivity Models (EVM) These models are known also as the first order models. They are based on the assumption that turbulent fluxes for species, momentum and heat depend directly on the mean–flow variables c_i, U_i and T.

Differential Second–Moment Closure (DSM/SCM) Models These models are described by Reynolds Stress/Flux transport equation, i.e. a separate modelled differential equation is solved for each turbulent flux $\overline{\rho}\widetilde{\varphi'' u_j''}$.

As VECTIS-MAX provides eddy viscosity $k-\varepsilon$ models, the eddy viscosity approach will be outlined next.

9.2.1 Eddy viscosity formulation

Eddy viscosity models have a root in an (in principle false) analogy between molecular and turbulent transport formulated by Bousinessq in 1877. Accordingly, Reynolds stresses can be obtained from the "constitutive" equation similar to Stokes law, Equation (8.2):

$$\tau_{ij}^{t} = -\rho\widetilde{u_i''u_j''} = 2\mu_t\left(S_{ij}-\frac{1}{3}S_{kk}\delta_{ij}\right)-\frac{2}{3}\rho k\delta_{ij} \tag{9.4}$$

where the mean–flow strain tensor S_{ij} is:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j}+\frac{\partial U_j}{\partial x_i}\right). \tag{9.5}$$

and μ_t is the *turbulent or eddy viscosity*. The term $2\rho k\delta_{ij}/3$ represents the mean turbulence pressure which is, together with velocity divergence term S_{kk}, usually added to the unknown static pressure of the resolved motion p_s by replacing it by the sum

$$p = p_s + \frac{2}{3}\left((\mu+\mu_t)S_{kk}+\rho k\right),\ S_{kk} = \frac{\partial U_k}{\partial x_k} \tag{9.6}$$

By using different arguments - analogy with the kinetic theory of gases, dimensional analysis and phenomenological models (cf. Speziale [1991]), the turbulent viscosity is defined in terms of characteristic turbulence length, time and velocity scales:

$$\mu_t = \rho\frac{\ell_t}{T_t}\ell_t = \rho v_t \ell_t\ , \tag{9.7}$$

Turbulent fluxes for species and heat can be formulated in a similar way by assuming an analogy between molecular and turbulent diffusion (see Equations (8.9) and (8.5)):

$$J_{c_i,j}^{t} = -\rho\widetilde{c_i''u_j''} = \mathscr{D}_{i,t}\frac{\partial c_i}{\partial x_j} \tag{9.8}$$

$$q_j^{t} = -\rho c_p\widetilde{T''u_j''} = \lambda_t\frac{\partial T}{\partial x_j} \tag{9.9}$$

with $\mathscr{D}_{i,t}$ and λ_t being turbulent (eddy) diffusivity for i–th species and turbulent thermal conductivity, respectively. The turbulent mass diffusivity and thermal conductivity are usually defined in terms of turbulent viscosity:

$$\rho\mathscr{D}_{i,t} = \frac{\mu_t}{Sc_{i,t}},\ \lambda_t = \frac{\mu_t c_p}{Pr_t} \tag{9.10}$$

where $Sc_{i,t}$ is the turbulent Schmidt (Prandtl) number for mass transport and Pr_t denotes turbulent Prandtl number for thermal transport. The above equation implies an analogy between the momentum, species and mass transfer. It is now possible to express the total diffusion fluxes (laminar plus turbulent) through so–called effective diffusivity values:

$$\mu_{eff} = \mu+\mu_t,\ \mathscr{D}_{i,eff} = \mathscr{D}_i+\mathscr{D}_{i,t},\ \lambda_{eff} = \lambda+\lambda_t \tag{9.11}$$

In Equation (9.4), the relation between Reynolds stresses and the mean strain tensor (i.e. the mean velocity gradient) is linear and the corresponding models are called *linear* EVM models.

Non–linear and *algebraic Reynolds stress* models have the Reynolds stress tensor defined as a general polynomial function of the mean velocity gradient, cf. Gatski and Rumsey [2002]. For example, models with the quadratic terms can be cast in the following form:

$$\tau_{ij}^{t} = -\frac{2}{3}\rho k\delta_{ij} + 2\mu_t\left(S_{ij} - \frac{1}{3}S_{kk}\delta_{ij}\right) + \mu_t\frac{k}{\varepsilon}\beta_1\left(S_{ik}S_{kj} - \frac{1}{3}S_{mn}S_{mn}\delta_{ij}\right)$$
$$+\mu_t\frac{k}{\varepsilon}\beta_2\left(S_{ik}W_{kj} + S_{jk}W_{ki}\right) + \mu_t\frac{k}{\varepsilon}\beta_3\left(W_{ik}W_{kj} - \frac{1}{3}W_{mn}W_{mn}\delta_{ij}\right)\,, \tag{9.12}$$

where β_i are closure coefficients and W_{ij} represents the mean rotation tensor:

$$W_{ij} = \frac{1}{2}\left(\frac{\partial U_i}{\partial x_j} - \frac{\partial U_j}{\partial x_i}\right). \tag{9.13}$$

In case of non–linear models, the closure coefficients are calibrated with the help of experimental or numerical data and physical constraints. For the algebraic stress models, the coefficients are derived in a consistent way from the full DSM models.

Depending on the number of differential equations which are used to predict turbulence scales in Equation (9.7), the EVM models can be classified as:

□ **Zero equation (algebraic) models.** Both turbulent length scale ℓ_t and time scale T_t are specified algebraically employing the empirical Prandtl mixing length theory. Examples are Smagorinsky [1963] and Baldwin and Lomax [1978] models.

□ **One equation models.** In these models one of turbulence scales is obtained from its own transport equation. The turbulent kinetic energy is usually used to define the velocity scale $v = k^{1/2}$, while the distribution of the length scale ℓ_t is prescribed empirically. In some models, the modelled transport equation for the turbulent viscosity μ_t is solved. An example is the Spalart–Almaras model (cf. Spalart and Allmaras [1994]) which has become popular in aeronautics.

□ **Two equation equation models.** Since Kolmogorov's pioneering work 1941 (cf. Spalding [1991]) , the turbulent energy k has been used without exception as the first variable in two- equation models.

- *$k-\omega$ models.* Kolmogorov proposed the two-equation model based on transport equations for the turbulent energy (actually two-thirds of k), and mean 'frequency' $\omega \propto k^{1/2}/\ell_t$. Although he did not explicitly provide the production term for ω - see Spalding [1991], his model inspired the development of modern $k-\omega$ models, Saffman [1970]; Wilcox [1998]; Menter [1994]. The ω variable can be interpreted as the ratio of ε and k, i.e. the rate of dissipation per unit of turbulent kinetic energy
- *$k-\varepsilon$ models.* Apart from k, Harlow and Nakayama [1967] introduced the turbulent dissipation rate $\varepsilon \propto k^{3/2}/\ell_t$ as the second turbulence scaling variable. The choice of ε as the second variable has been the most popular: it appears in the k-equation as the sink term, and its exact transport equation can be easily derived from the Navier-Stokes equations. Unfortunately, modelling of the ε- equation is extremely difficult, and in the past relied on intuition combined with dimensional consistency, coordinate invariance, and analogy with k-equation. Based on the work of Davydov [1961], Hanjalic [1970] (see also Hanjalic and Launder [1972]) proposed a simple modelled ε-equation for high Re-number flows. He also established model coefficients that differ a little from

values proposed by Jones and Launder [1972]; Launder and Spalding [1974]. The model published by Launder and Spalding [1974] is often referred to as the *standard* $k-\varepsilon$ *model*. Some weaknesses of the standard model can be removed by using the RNG $k-\varepsilon$ (cf. Yakhot and Orszag [1986]; Yakhot et al. [1992]) and realisable model of Shih et al. [1995]. The term realisable describes satisfaction of certain mathematical constraints so that generation of physically unrealistic variable values, e.g. negative values of k and its components (normal turbulent Reynolds stresses), is prevented. Neither standard nor RNG models are realisable.

– *Shear Stress Transport (SST) model.* This model, developed by Menter [1994], is formulated as the $k-\omega$ model but effectively it is a combination of the $k-\varepsilon$ and $k-\omega$ models. Close to the wall, it reduces to the $k-\omega$ model and away from the wall to the $k-\varepsilon$ model. It was claimed to perform better than the Wilcox $k-\omega$ and $k-\varepsilon$ models, especially for separated and adverse pressure gradient flows.

□ **Advanced models.** This group of models provides some advanced modelling features not available in the above *linear* two–equation models. It includes

– *Non–linear and Algebraic Reynolds stress models.* Some critical deficiencies of the linear $k-\varepsilon$ models can be rectified by defining non-linear relation between Reynolds stresses and the mean velocity gradients, see Equation (9.12).

– *Elliptic–relaxation models.* Durbin [1991] developed the Elliptic relaxation model, referred to as the $k-\varepsilon-v^2$ model, which is probably the most successful EVM model for near–wall turbulence modelling. Apart from k and ε equations, two additional equations are solved: the equation for wall–normal Reynolds stress v^2 and for the elliptic relaxation function.

9.3 Linear Two–Equation $k-\varepsilon$ Models

Together with the mean flow equations (6.28–6.31) , transport equations for the turbulent kinetic energy k and its dissipation rate ε are solved in the $k-\varepsilon$ models. Replacing the time and length scale in Equation (9.7) by turbulent scales from Equation (9.2), the turbulent viscosity can be defined as

$$\mu_t = \rho C_\mu f_\mu k T_t,\ T_t = max\left(\frac{k}{\varepsilon}, C_t\sqrt{\frac{\mu}{\rho\varepsilon}}\right) \tag{9.14}$$

where C_μ and C_t are model coefficients and f_μ is the near–wall dumping function, employed in conjunction with low–Reynolds number models. In the above equation for T_t, the turbulence time scale k/ε is bounded from below by the Kolmogorov time scale $T_k = C_t\sqrt{\mu/(\rho\varepsilon)}$, as proposed by Durbin [1991]; Yang and Shih [1993].

In order to close the diffusion term D_k in equation of k (6.40), one can assume that its spatial gradient drives the turbulent transport by triple product of velocity fluctuations $\overline{\rho}\widetilde{k''u_j''} = \overline{\rho u_i'' u_i'' u_j''}/2$. The same assumption is physically unsound for the transport by pressure fluctuations, Bradshaw [1994]. Since it appears that the turbulent transport by the velocity fluctuations is dominant one, modelling of both terms by a gradient transport hypothesis:

$$D_k = \frac{\partial}{\partial x_j}\left(\overline{\tau_{ij}' u_i'} - \overline{\rho}\widetilde{k''u_j''} - \overline{p'u_j'}\right) = \frac{\partial}{\partial x_j}\left(\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right) , \tag{9.15}$$

can be considered as an intentional approximation. A non–dimensional constant $\sigma_k \approx 1$ is the effective Prandtl number for the diffusion of k. Turbulent kinetic energy production by shear is calculated as:

$$P_k = \tau_{ij}^t \frac{\partial U_i}{\partial x_j} = \mu_t \left(S^2 - \frac{2}{3} S_{kk}^2 \right) - \frac{2}{3} \rho k S_{kk} \tag{9.16}$$

where S represents the strain tensor invariant

$$S = \sqrt{2 S_{ij} S_{ij}} \tag{9.17}$$

Modelling of the production by body force, P_b, depends on the character of the force. The gravitational force $f_i = g_i$ is frequently encountered and for flows driven by thermal or concentration buoyancy the production term P_b becomes $\rho \widetilde{f_i'' u_i''} = -g_i \overline{\rho' u_i'}$. Following the standard eddy–diffusivity approach, the unknown correlation $\overline{\rho' u_i'}$ is modelled by the gradient of mean–flow density:

$$P_b = -\frac{\mu_t}{\rho Pr_\rho} g_i \frac{\partial \rho}{\partial x_i} \tag{9.18}$$

where Pr_ρ is usually taken to be the same as for the temperature, i.e. $Pr_\rho \approx Pr_t$. The compressibility correction terms in the k–equation have been neglected having in mind their current state of knowledge, Leschziner et al. [2000]; Barre et al. [2002].

The dissipation rate of turbulence kinetic energy, $\varepsilon = \overline{\tau_{ij}' \partial u_i' / \partial x_j} / \rho$, appears as the sink term in the k–equation. While an exact transport equation for ε can be derived and interpreted in the physical terms (cf. Speziale [1996]), the model equation relies much more on the physical and empirical reasoning than on the rigorous modelling of the exact equation.

An important issue regarding the modelled and exact equation is that the dissipation rate used in the definition of the eddy–viscosity, Equation (9.14), actually represents the energy transfer rates from large energy–containing eddies. This quantity is generally different than ε given by the exact equation, describing the dissipative process in the small eddies.

9.3.1 Standard $k-\varepsilon$ model

The following model equations for the turbulent kinetic energy and its dissipation rate describe the basic (standard – SKE) $k-\varepsilon$ model (cf. Hanjalic [1994]; Speziale [1996]):

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_j}\left(\rho k \left(U_j - U_{g,j}\right)\right) = P_k + P_b - \rho \varepsilon + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \tag{9.19}$$

$$\begin{aligned} \frac{\partial}{\partial t}(\rho \varepsilon) + \frac{\partial}{\partial x_j}\left(\rho \varepsilon \left(U_j - U_{g,j}\right)\right) &= \frac{C_{\varepsilon 1} P_k - C_{\varepsilon 2} \rho \varepsilon + C_{\varepsilon 3} P_b}{T_t} \\ & -C_{\varepsilon 4} \rho \varepsilon S_{kk} + \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + S_\varepsilon \end{aligned} \tag{9.20}$$

9.3.2 RNG $k-\varepsilon$ model

The Renormalisation Group Theory (RNG) applied to turbulence modelling by Yakhot and Orszag [1986] and Yakhot et al. [1992] offered an alternative RNG $k-\varepsilon$ model. Compared to the basic $k-\varepsilon$ model, the ε–equation has now an additional production term $S_\varepsilon = -\mathscr{R}$

$$\mathscr{R} = \frac{\eta\left(1-\frac{\eta}{\eta_0}\right)}{1+\beta\eta^3}\frac{\varepsilon}{k}P_k, \ \eta = S\frac{k}{\varepsilon} \tag{9.21}$$

where $\eta_0 = 4.38$ and $\beta = 0.012$.

9.3.3 Standard $k-\varepsilon$ model with the realisable time–scale bound (TSB)

Imposing the realisability constraint on the eddy–viscosity relation for the Reynolds stresses, Equation (9.4), written in the principal axes of S_{ij}, Durbin [1996] derived an upper bound on the turbulent time scale T_t:

$$T_r = \frac{\alpha_r}{\sqrt{3}C_\mu S} \tag{9.22}$$

where the constant $\alpha_r = 1$. Combining this upper bound and the lower bound pertinent to the Kolmogorov time scale, the turbulent time scale Equation (9.14) becomes:

$$T_t = \max\left(\min\left(\frac{k}{\varepsilon},\frac{1}{\sqrt{3}C_\mu S}\right), C_t\sqrt{\frac{\mu}{\rho\varepsilon}}\right) \tag{9.23}$$

The realisability constraint on the time scale ensures that normal Reynolds stress components τ^t_{ii} are always positive.

9.3.4 The $k-\varepsilon$ model coefficients

The model coefficients appearing in the above $k-\varepsilon$ equations are assigned the values given in Table 9.1.

Table 9.1: Model coefficients for the $k-\varepsilon$ models.

Model	C_μ	σ_k	σ_ε	$C_{\varepsilon 1}$	$C_{\varepsilon 2}$	$C_{\varepsilon 3}$	$C_{\varepsilon 4}$	C_t	α_r	η_0	β	Pr_t
SKE(R)	0.09	1.0	1.3	1.44	1.90	$(\lvert P_b\rvert + P_b)/2$	$2(2-C_{\varepsilon 1})/3$	1.	1.	–	–	0.9
RNG	0.0845	0.72	0.72	1.42	1.68	$(\lvert P_b\rvert + P_b)/2$	$2(2-C_{\varepsilon 1})/3$	–	–	4.38	0.012	0.9

9.4 Near–Wall Modelling: Wall Functions

The $k-\varepsilon$ models presented so far are of the 'high Reynolds number' type in that they are not applicable in the near–wall region. More precisely, they can't account for viscous and wall blocking effects. In order to account for these effects, they ought to be either used with

- *Wall functions*: formulae that describe the flow variables within the fully turbulent (logarithmic) layer and bridge the viscous– affected layers,
- or re–formulated as *Low–Reynolds number models* that can be integrated up to the wall.

As a very dense numerical mesh is required to resolve the near–wall region, the wall functions require significantly less of computing time than the low *Re*–number models.

The log–law velocity profile forms the basis of the the standard 'wall–function' approach. The purpose of this approach is to link the solution variables at near–wall cells, placed within the log–law layer, with those at the wall. However, the key difficulty is provision of the desirable mesh clustering near the walls, especially in case of automatic mesh generation. For example, the Cartesian cut–cell grid generators always produce the near–wall grids with very different cell volumes. Therefore the first grid points (i.e. the near-wall cell centres) can lie within either viscous sub–layer or buffer layer or logarithmic law region. This means that any type of wall functions have to provide formulae for the solution variables at arbitrary placed near–wall cells.

Three types of wall functions are available in VECTIS-MAX :

- Standard or conventional,
- Scalable and
- Unified or enhanced wall functions.

To understand them better, some physical aspects of the near–wall region, including a logarithmic law of the wall, are introduced.

9.4.1 Background: logarithmic law

The near–wall region is characterised by high velocity gradients and dominant molecular effects. In essence, the wall modifies the mean flow and the turbulence in its vicinity through *viscous* as well as *non–viscous* effects. Viscous effects reduce the velocity fluctuations parallel to the wall. Non–viscous effects are due to the kinematic blocking of the velocity fluctuations normal to the wall (cf. Durbin [1991]). Also, the fluctuating pressure field is modified by the presence of the wall.

The *friction velocity* U_τ and velocity scale U_k, defined as

$$U_\tau = \sqrt{\frac{|\tau_w|}{\rho}}\ ,\ U_k = C_\mu^{1/4} k^{1/2} \tag{9.24}$$

(τ_w is the wall shear stress) are commonly used as wall normalisation variables. Thus the velocity parallel to the wall (a non–moving wall is assumed) and the normal distance from the wall, denoted as y, can be normalised either by U_τ or U_k. Normalisation by U_τ gives the *wall or plus units*:

$$U^+ = \frac{U}{U_\tau}\ ,\ y^+ = \frac{\rho U_\tau y}{\mu} \tag{9.25}$$

while normalisation with the k–based velocity scale U_k gives the *star units*:

$$U^* = U^+ \frac{U}{U_k} = \frac{\rho U U_k}{\tau_w} \ , \ y^* = \frac{\rho U_k y}{\mu} = \frac{\rho C_\mu^{1/4} k^{1/2} y}{\mu} \tag{9.26}$$

Numerous experimental and DNS data as well as dimensional analysis suggest that the inner part of the near–wall region comprises three layers:

- *Viscous sub-layer*: the flow is dominated by the viscous force represented by viscosity.
- *Buffer layer*: Both molecular and turbulence effects are important.
- *Fully turbulent or logarithmic law layer*: Turbulence (inertial force) plays a dominant role.

As Figure 9.1 shows, the distribution of the non–dimensional velocity parallel to the wall $U^+ = f(y^+)$ is governed by distinctive laws throughout the viscous and fully turbulent layers. In the viscous sub-layer, the velocity obeys the linear law while the so–called *universal velocity profile* or *log–law* describes the fully turbulent layer:

$$U^+ = \begin{cases} y^+ \ , & \text{viscous sub-layer: } y^+ \leq 5 \\ \frac{1}{\kappa}\ln\left(E y^+\right) \ , & \text{log-law: } y^+ \geq (30-50) \end{cases} \tag{9.27}$$

where $\kappa \approx 0.41$ is the von Karman constant, while E is another empirical constant whose value depends on the wall roughness; for smooth walls $E \approx 9$.

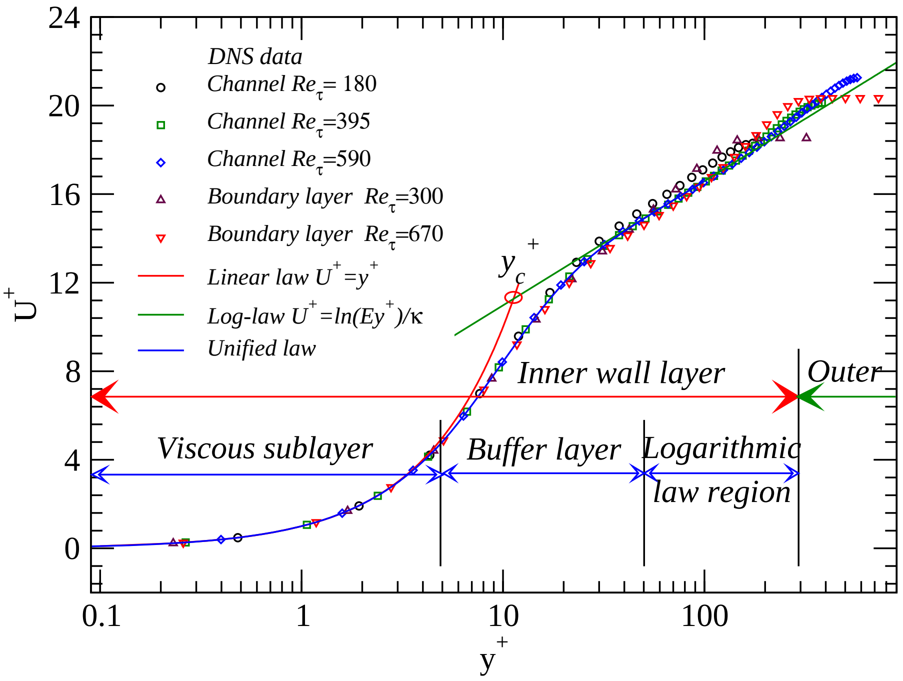


Figure 9.1: Velocity distribution in the inner wall region for channel and boundary layer flows with zero pressure gradient.

The log–law appears to be valid* for simple boundary layer flows in local equilibrium, that is when the production of the turbulent kinetic energy balances its dissipation, $P_k = \rho\varepsilon$. In terms of Prandtl's mixing–length theory, the log–law can be obtained by assuming that the turbulence length scale ℓ_t in Equation (9.7) is proportional to the normal distance from the wall y: $\ell_t = \kappa y$. For a turbulent layer in local equilibrium, the following relations can be derived (say at point 'P' within the turbulent layer):

$$U_\tau = C_\mu^{1/4} k_P^{1/2} \text{ or } k_P = \frac{U_\tau^2}{\sqrt{C_\mu}} , \tag{9.28}$$

$$\varepsilon_P = \frac{U_\tau^3}{\kappa y_P} = \frac{C_\mu^{3/4} k_P^{3/2}}{\kappa y_P} , \tag{9.29}$$

$$\mu_t = \rho\kappa U_\tau y_P = \rho\kappa C_\mu^{1/4} k_P^{1/2} y_P . \tag{9.30}$$

9.4.2 Standard wall functions

Launder and Spalding [1974] re–defined the log–law in terms of the velocity scale U_k ('star' units) as follows:

$$U_P^+ = \frac{U_\tau}{\kappa U_k} \ln\left(E y_P^*\right) \quad \text{or} \quad U_{P,log}^* = \frac{\rho U_P U_k}{\tau_w} = \frac{1}{\kappa} \ln\left(E y_P^*\right) \tag{9.31}$$

This formulation relaxes a condition where $\tau_w \to 0$ and for $U_k/U_\tau = 1$ the universal log–law, Equation (9.27), is recovered. The ratio (U_k/U_τ) can be seen as a non–equilibrium index that changes the slope of the velocity profile in the turbulent layer (cf. Kim and Choudhury [1995]). It can also takes into account some non–equilibrium departures, i.e. when the production of k is not in balance with its dissipation ε. Thus, the above equation is used to calculate explicitly the wall shear stress:

$$\tau_w = \frac{\rho\kappa U_k}{\ln(E\, y_P^*)} U_P \tag{9.32}$$

If the wall moves with velocity $\vec{U}_w$ then the velocity U_P in all the above equations should be replaced by the difference of velocities parallel to the wall surface, ΔU_P:

$$\Delta U_P = \sqrt{\left(\vec{U}_w - \vec{U}_P - \left[\left(\vec{U}_w - \vec{U}_P\right)\cdot\vec{n}_w\right]\vec{n}_w\right)^2} \tag{9.33}$$

where $(\vec{U}_w - \vec{U}_P)\cdot\vec{n}_w$ denotes the magnitude of velocity normal to the wall and $\vec{n}_w$ is the outside wall surface normal.

Using similarities between the momentum and energy transfer, and neglecting the viscous heating, the normalised temperature, given as

$$T^* = \frac{(T_w - T_P)\rho c_p U_k}{q_w} \tag{9.34}$$

can be also approximated with a log–law expression. However, it is commonly expressed in terms of the normalised velocity U^*:

$$T_{log}^* = Pr_t\left[U^* + P_Y\left(\frac{Pr}{Pr_t}\right)\right] \tag{9.35}$$

* The on–going debate, cf. George [2007], questions existence of the universal log–law for wall-bounded flows

The function P_Y represents the viscous thermal resistance and according to Jayatillaka [1969] it reads:

$$P_Y = 9.24\left[\left(\frac{Pr}{Pr_t}\right)^{3/4} - 1\right]\left[1 + 0.28\exp\left(-0.007\frac{Pr}{Pr_t}\right)\right] \tag{9.36}$$

The distribution of species $c_i^* = f(y^*)$ in the fully turbulent layer is described analogously to the temperature:

$$c_i^* = \frac{(c_{i,w} - c_{i,P})\,\rho U_k}{J_{i,w}} \tag{9.37}$$

$$c_{i,log}^* = Sc_t\left[U^* + P\left(\frac{Sc}{Sc_t}\right)\right] \tag{9.38}$$

where $J_{i,w}$ is the diffusion flux of species i at the wall; the function $P(Sc/Sc_t)$ is calculated from Equation (9.36) by replacing molecular and turbulent Prandtl numbers (Pr and Pr_t) with corresponding Schmidt numbers Sc and Sc_t, respectively.

Numerically, the wall fluxes (shear stress, heat flux and diffusion flux of species) can be defined in terms of the wall effective viscosity μ_w, thermal conductivity λ_w and mass diffusion coefficient $\mathscr{D}_{i,w}$, respectively:

$$\tau_w = \mu_w\frac{\Delta U_P}{y_P},\ q_w = \lambda_w\frac{(T_w - T_P)}{y_P},\ J_{i,w} = \rho\mathscr{D}_{i,w}\frac{(c_{i,w} - c_{i,P})}{y_P} \tag{9.39}$$

from which, and Equations (9.32, 9.34, 9.37), the effective wall diffusion coefficients follow as:

$$\mu_w = \mu\frac{y_P^*}{U_P^*},\ \lambda_w = \lambda\frac{Pry_P^*}{T_P^*},\ \mathscr{D}_{i,w} = \mathscr{D}_i\frac{Scy_P^*}{c_{i,P}^*} \tag{9.40}$$

As before, μ, λ and $\mathscr{D}_i$ are molecular diffusion coefficients for momentum, heat and mass transfer, respectively. In the viscous sub–layer linear laws apply

$$U_{vis}^* = y_P^*,\ T_{vis}^* = Pr\,y_P^*\ ,\ c_{i,visc}^* = Sc\,y_P^* \tag{9.41}$$

so that the wall effective coefficients reduce to the molecular ones.

The standard wall function approach excludes the buffer layer by extending the viscous and turbulent layers up to their point of intersection, $y_c^* = 11.63$, see Figure 9.1. The following formulae describe the distribution of velocity, temperature, mass fraction of species and turbulence variables.

□ *Velocity and wall shear stress.*

$$U_P^* = \begin{cases} y_P^*\ , & \text{viscous sub-layer: } y_P^* < y_c^* \\ \frac{1}{\kappa}\ln(Ey_P^*)\ , & \text{log-law: } y_P^* \geq y_c^* \end{cases} \tag{9.42}$$

The wall shear stress is calculated from Equation (9.39) with the wall viscosity obtained from Equation (9.40).

- *Temperature and wall heat flux.* For the entire wall region a unified expression is obtained by using a blending model of Kader [1981]:

$$T_P^* = e^{-\Gamma} T_{vis}^* + e^{-1/\Gamma} T_{log}^* \tag{9.43}$$

where Kader's blending function is given as:

$$\Gamma = \frac{0.01\,(Pr y_P^*)^4}{1 + 5Pr^3 y_P^*} \tag{9.44}$$

The wall heat flux is given by Equation (9.39) with the wall thermal conductivity calculated from Equation (9.40).

- *Mass fraction and wall diffusion flux of species.* Formulae used for the temperature and heat flux apply also to species if the Prandtl numbers are replaced by corresponding Schmidt numbers.

- *Turbulent energy production.* The k-equation is solved for the near–wall cells with zero value and zero diffusion flux at the wall. Its production within the log–law layer can be calculated with respect to the velocity distribution given by Equation (9.31):

$$P_{k,P} = \begin{cases} \tau_P^t \left(\frac{\partial U}{\partial y}\right)_P = \frac{\tau_w^2}{\mu \kappa y_P^*}\,, & y_P^* \geq y_\varepsilon^* \\ 0 & y_P^* < y_\varepsilon^* \end{cases} \tag{9.45}$$

where the turbulent stress is approximated by the wall shear stress ($\tau_P^t \approx \tau_w$). The constant shear stress τ_w can be assumed across the entire near–wall region. The value of $y_\varepsilon^* = 2\kappa/C_\mu^{1/2} \approx 2.733$ corresponds to the intersection of equations defining the dissipation rate in the log–law layer ε_{log}, (Equation (9.29)), and in the wall limit ε_{vis}:

$$\varepsilon_{log} = \frac{C_\mu^{3/4} k_P^{3/2}}{\kappa y_P}\,, \varepsilon_{vis} = \frac{2\mu k_P}{\rho y_P^2} \tag{9.46}$$

- *Turbulent energy dissipation rate.* In the near-wall cells the dissipation rate is set explicitly to:

$$\varepsilon_P = \varepsilon_{log} \frac{1}{1 - e^{-\gamma Re_{k,P}}}\,,\; Re_k = \frac{\rho k^{1/2} y}{\mu} \tag{9.47}$$

The above expression, covering the entire wall region, was proposed by Wolfshtein [1969] with $\gamma = 0.263$. To get a correct value in the wall limit ε_{vis}, the parameter $\gamma = 0.2$ needed to be used, Rung et al. [2000].

9.4.3 Scalable wall functions

The scalable wall functions, Esch and Menter [2003], attempt to remove inconsistencies with the standard ones caused by the non–uniform near–wall grid resolution. This is achieved by limiting the normalised wall distance y^* from below by $y_c^* = 11.63$. The above standard wall functions are therefore used in which $y_{P,lim}^* = \max(y_P^*, y_c^*)$ replaces y_P^*. The y^*–limiting artificially moves near–wall cells from the viscous sub–layer into the turbulent layer.

9.4.4 Enhanced (unified) wall functions

Enhanced wall functions, described in Przulj [2009], have been designed in terms of non–dimensional wall distances y^* ('star' units) in order to provide a smooth distribution of flow variables across viscous sub-layer, buffer layer and logarithmic law region. In the viscous sub-layer they satisfy corresponding wall–limiting expressions and in the fully turbulent region they are identical to the standard wall functions,

- ☐ *Velocity distribution.* Figure 9.2 shows the scaling of the channel and boundary layer DNS data (Moser et al. [1999]; Spalart [1988]) in a form $U^* = F(y^*)$. All data collapse well for $y^* < 15$

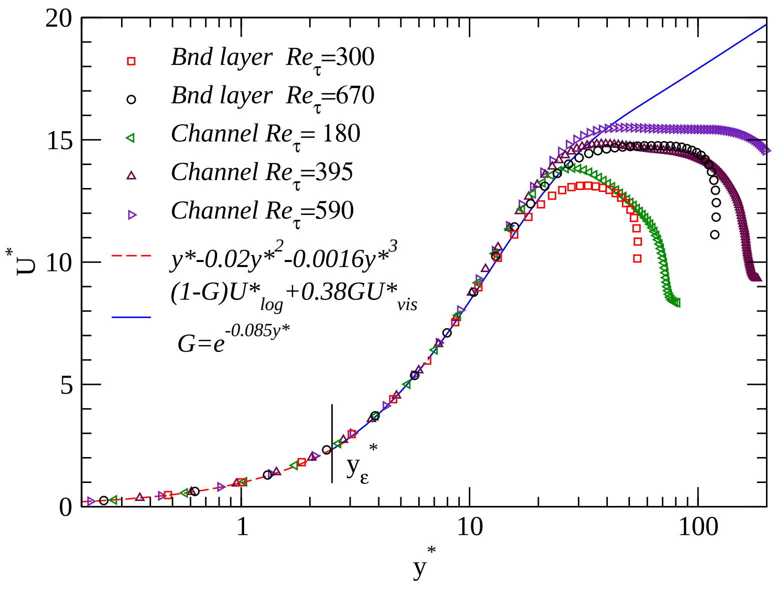


Figure 9.2: A–priory comparison of $U^* = F(y^*)$ expression (9.49) with DNS data for boundary layer Spalart [1988] and channel Moser et al. [1999] flows.

and the linear relationship, Equation (9.41), holds within the viscous sub–layer ($y^* \leq y^*_\varepsilon \approx 2.5$). The approximate validity of the log–law Equation (9.31) is restricted to a narrow range of data for $15 \leq y^* \leq 30-40$. Notably, classical scaling $U^+ = F^+(y^+)$, shown in Figure 9.1, results with much wider log–law region, $30 \leq y^+ \leq 200$.

With the help of the blending function

$$G = \exp(-\alpha_u y^*), \ \alpha_u = 0.085 \approx C_\mu \tag{9.48}$$

the DNS data are correlated in terms of 'star' units as follows:

$$U^* = F(y^*) = \begin{cases} y^* - 0.02y^{*2} - 1.6\times10^{-3}y^{*3} \ , & \text{viscous sub-layer: } y^* < y^*_\varepsilon \\ (1-G)\ln(Ey^*)/\kappa' + \kappa' G y^* \ , & \text{buffer \& log-law: } y^* \geq y^*_\varepsilon \end{cases} \tag{9.49}$$

where $\kappa' = 0.38$ and y^*_ε is the intersection point between the wall limiting and fully turbulent ε–expressions, see Equation (9.46). The blended profile in the above equation can also be used within the viscous sub–layer. However, the third order polynomial is introduced instead as it correlates better the DNS data than the blended profile. A blending function $G = \exp(-\alpha_u^+ y^+), \alpha_u^+ = 0.1$ is used to define a unified law depicted in Figure 9.1.

- *Temperature and mass fraction of species.* For the entire wall region the unified expressions presented for the standard wall functions are employed, see Equations (9.43, 9.44).

- *Turbulent energy production.* Following the standard wall function practice, the near–wall production of turbulent energy is calculated using the velocity gradient obtained from the Equation (9.49). In addition, the constant shear stress τ_w across the near–wall region is assumed. Bearing in mind that $k = k(y)$, the velocity gradient can be written as:

$$\frac{\partial U}{\partial y} = \frac{\partial \left(U^* \, U_\tau^2/U_k\right)}{\partial y^*}\frac{\partial y^*}{\partial y} = \frac{\tau_w}{\mu} F_1(y^*, y), \tag{9.50}$$

$$F_1(y^*, y) = \frac{\partial U^*}{\partial y^*} + \frac{y}{2k}\frac{\partial k}{\partial y}\left(\frac{\partial U^*}{\partial y^*} - \frac{U^*}{y^*}\right) \tag{9.51}$$

Neglecting the normal–stress production, the production of k becomes:

$$P_k = \tau_t \frac{\partial U}{\partial y} = \frac{\tau_w^2}{\mu}\left[1 - F_1(y^*, y)\right] F_1(y^*, y) \tag{9.52}$$

where $\tau_t = \tau_w - \mu(\partial U/\partial y)$ denotes the turbulent near-wall shear stress.

The exact form of the function F_1, Equation (9.51), is not used currently. Instead, a form of $F_1(y^*, y) \approx (\partial U^*/\partial y^*)$ is assumed. To compensate for neglected $\partial k/\partial y$ derivative in F_1, the coefficient α in the blending function (9.48) has been re–tuned with the help of DNS data to a value of $\alpha_k = 0.075$. Also, κ' has been set to 0.37 and the final piecemeal function is given as:

$$F_1(y^*) = \begin{cases} 1 - 0.008y^* - 0.005y^{*2}\,, & y^* < y^*_\varepsilon \\ (1-G)/(\kappa' y^*) + G\left[\kappa'(1 - 0.09y^*) + 0.075\ln(Ey^*)/\kappa'\right]\,, & y^* \geq y^*_\varepsilon \end{cases} \tag{9.53}$$

As can be seen in Figure 9.3 (left), the expression for normalised turbulent energy production P_k^+, defined as $P_k^+ = P_k\mu/\tau_w^2 = [1 - F_1(y^*)]F_1(y^*)$ shows satisfactory agreement with the DNS data for channel flows.

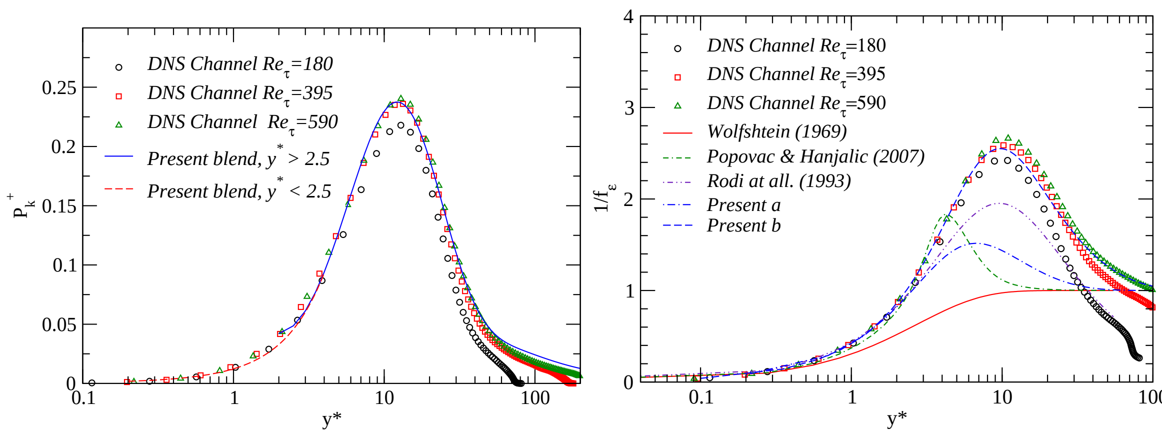


Figure 9.3: A–priory comparison of wall functions for near–wall turbulent energy production (left) and its dissipation rate (right), Eqs. (9.52) and (9.55, 9.56), respectively, with DNS data for channel flows, Moser et al. [1999].

□ *Turbulent energy dissipation rate.* The blending function (9.48), with $\alpha = \alpha_u$, is also employed to define the turbulence energy dissipation rate ε:

$$\varepsilon = G\varepsilon_{vis} + (1-G)\,\varepsilon_{log}, \tag{9.54}$$

where the viscous (wall limit) and log–law (fully turbulent) values, denoted as ε_{vis} and ε_{log}, respectively are given by Equation (9.46). Note that the y^*_ε value represents the intersection point between the wall limiting and fully turbulent ε–expressions. Introducing the function $f_\varepsilon(y^*) = \varepsilon/\varepsilon_{log}$, Equation (9.54) can be expressed as:

$$\varepsilon = f_\varepsilon \varepsilon_{log}, \;\; f_\varepsilon = 1 - G\left(1 - \frac{y^*_\varepsilon}{y^*}\right) \tag{9.55}$$

Figure 9.3 (right) compares the above ε–blending function (labelled as 'Present a') against DNS data. Unified expressions proposed by Wolfshtein [1969]; Rung et al. [2000], Rodi et al. [1993] and Popovac and Hanjalic [2007] are also evaluated. Large differences between the considered expressions and between each individual expression and DNS data are evident. Another expression, denoted as 'Present b', and having a similar form as Equation (9.55), is in good agreement with DNS data. This expression is formulated as:

$$f_\varepsilon = 1 - \exp(-0.031y^*) + \frac{y^*_\varepsilon}{y^*}\exp(-0.07y^*) \tag{9.56}$$

and currently used to calculate near–wall dissipation rate according to Equation (9.55).

9.5 Low–Reynolds number modelling

The low–Reynolds $k-\varepsilon$ models have been devised as a synergy between the low Reynolds number model of Yang and Shih Yang and Shih [1993] and imposed physical bounds on the turbulence time scale pioneered by Durbin Durbin [1991, 1996, 2009]. Yang and Shih defined the dumping function f_μ in Equation (9.14) in terms of the mean strain rate tensor magnitude S (see Equation (9.17))

$$f_\mu = \sqrt{1 - \exp\left(-a_1R - a_2R^2 - a_3R^3\right)}, \; R = \frac{\rho k}{S\mu} \tag{9.57}$$

where the a_i coefficients read as: $a_1 = 3\times10^{-4}$, $a_2 = 6\times10^{-5}$ and $a_3 = 2\times10^{-6}$. This makes the model sensitive to local strain rate and removes ambiguity associated with dumping functions using wall normal distances. Away from the wall, the model reduces to the standard one as it employs the standard model constants. Another of the model's feature is the limiting of the turbulence time scale T_t from below by the Kolmogorov time scale T_k, Eq. (9.1), i.e. $T_t = k/\varepsilon + T_k$.

As reported by Watterson et al. [1999]; Yao et al. [2002], a truncated version of the model, where the additional source term S_ε in Equation (9.20) is neglected, has produced satisfactory results for separating flows (including vortex shedding) in complex geometries. Thus the same truncated version of the model is also employed here.

The low–Reynolds variants of either Standard $k-\varepsilon$ model (SKE) or Standard $k-\varepsilon$ model with realisable time–scale bound (TSB) are available. In the case of the SKE model, the turbulent time scale is bounded from below by the Kolmogorov time scale, Eq. (9.14). For the TSB model, in addition to the lower bound, Durbin's Durbin [1996] realisability condition is imposed as the upper bound on the time scale, see Eq.(9.23).

9.6 Inherent Limitations of $k-\varepsilon$ Models and Wall Functions

The linear $k-\varepsilon$ models available in VECTIS-MAX (and other two–equation EVM models) represent the simplest and *complete* turbulence closure based on the linear relationship between the Reynolds stress tensor and local mean strain rate tensor - Equation(9.4), where the proportionality coefficient is a scalar quantity - the turbulent viscosity μ_t. While this simple framework is computationally very efficient it has some purely physical shortcomings. The relation (9.4) can be expressed through the anisotropy stress tensor b_{ij}:

$$b_{ij} = -\frac{\tau^t_{ij} + 2\rho k \delta_{ij}/3}{2\rho k} = -\frac{\nu_t}{k}\left(S_{ij} - \frac{1}{3}S_{kk}\delta_{ij}\right) \tag{9.58}$$

In reality, the stress anisotropy is far from what is predicted by linear models. This and other deficiencies can be summarised as follows (cf. Hanjalic [1994]):

Linear Reynolds stress–strain relationship: A consequence is the isotropic eddy viscosity. Poor predictions for flows with turbulent stress transport, for example flows involving strong separation and buoyancy.

Insensitivity to stress anisotropy: Poor predictions wherever normal stresses are important, e.g. stress-driven secondary flows in the straight non-circular ducts cannot be predicted at all.

Insensitivity to extra strain: No physical mechanism to deal with complex flows (three-dimensional) characterised by streamline curvature, swirl and rotation.

However, the $k-\varepsilon$ models have shown remarkably good agreement with experimental data for simple flows (two–dimensional attached boundary layers, channels with pressure gradients) and even for some recirculating flows dominated by pressure gradients or flows with infirm streamline curvature.

Considering the conventional wall functions, they have been derived for simple boundary layer, near–equilibrium flows. It is very unlikely that wall functions will hold in complex flows. The same conclusion can be made for the scalable wall functions. They are expected to improve convergence properties of the solution method. The unified wall functions can potentially reduce the sensitivity of predicted results (i.e. remove the grid dependence of the results). Again, it is unlikely that these functions, used with the current high Reynolds number models, will significantly reduce the grid dependence of the results. For this, the near–wall models that can be integrated up the wall and properly account for the viscous and kinematic blocking wall effects should be used, see Popovac and Hanjalic [2007]. The current low–Reynolds number model variants provide integration up to the wall.

The characteristic flow areas such as the stagnation region, the boundary layers, the shear layers and the wake, shown in Figure 9.4 for the flow around a circular cylinder, are commonly found in typical engineering flow configurations. For such flows, the $k-\varepsilon$ models (and most of advanced

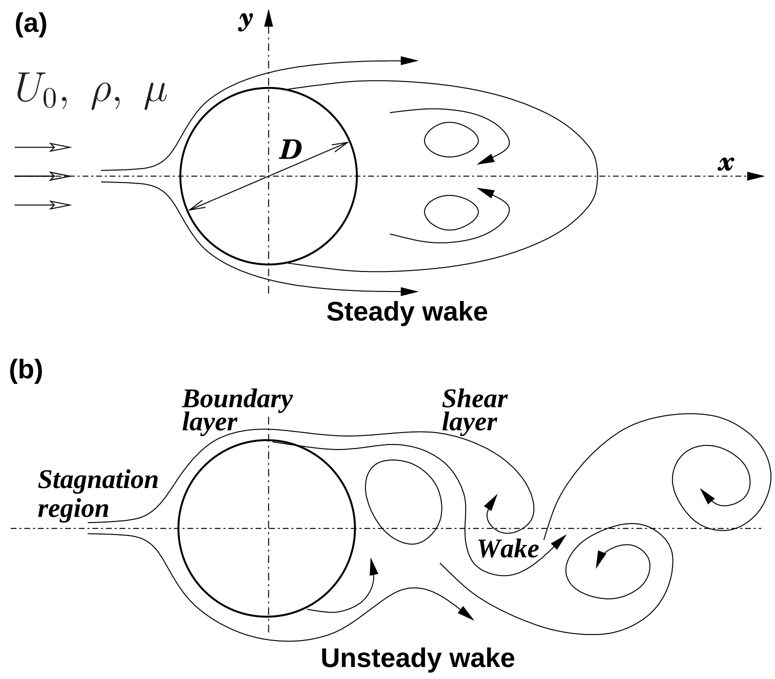


Figure 9.4: Characteristic flow regions for a circular cylinder: (a) steady separation, (b) unsteady separation (vortex shedding).

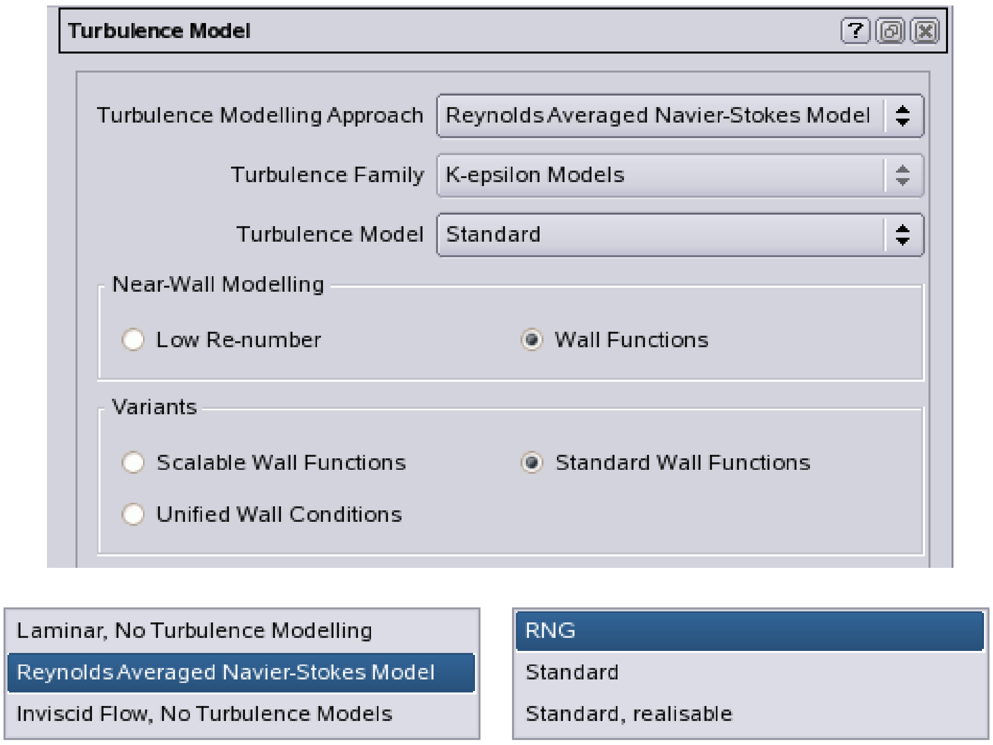
EVM models), used in conjunction with the wall functions, cannot simulate the transition to turbulence in the laminar boundary layers and/or predict accurately the points of flow separation and re-attachment. There is limited scope to improve the predictions in the stagnation regions by using either the RNG or Standard $k-\varepsilon$ model with the realisable turbulence time–scale bound (TSB). Further improvements can be expected if the low–Reynolds number variants of the standard and TSB model are employed on the refined grids.

Compared to the standard $k-\varepsilon$ model, the RNG based model improves predictions of separated flows, in particular the length of recirculating regions. However, it might spoil predictions of accelerating flows. It generally performs better in the flow regions where the normal stresses govern the production of turbulence. In these regions, usually associated with the stagnation point flow, the standard $k-\varepsilon$ model grossly over-predicts the turbulent kinetic energy and thus the level of turbulent viscosity. Transported downstream, this excessive level of k can seriously affect the accuracy of model predictions. Typical examples include the flow around a circular or square cylinder and impinging jet flow. The $k-\varepsilon$ model with the realisable turbulence time–scale bound (TSB) prevents excessive levels of the turbulent kinetic energy in the stagnation flow regions by sensibly amplifying the production of ε in these regions. It also improves predictions of the low turbulence level flows with a large time scale k/ε.

Having in mind the simplicity, robustness and computational economy, the implemented $k-\varepsilon$ models, and generally all eddy–viscosity models, are extremely useful tool for industrial flow calculations, provided that the users understand their limitations. In this way, potentially misleading and wrong results can be detected.

9.7 Setting a Turbulence Model and Wall Treatment

For each fluid domain, the turbulence modelling inputs have to be selected in the Turbulence Model panel. This panel is opened by left–click on the Turbulence Model node within the parent fluid domain node, see Figure 7.2. By default, the Turbulent Modelling Approach is set to a turbulent flow


Turbulence Model

Turbulence Modelling Approach: Reynolds Averaged Navier-Stokes Model
Turbulence Family: K-epsilon Models
Turbulence Model: Standard

Near-Wall Modelling: ○ Low Re-number [● Wall Functions]

Variants: ○ Scalable Wall Functions [● Standard Wall Functions] ○ Unified Wall Conditions

Laminar, No Turbulence Modelling
Reynolds Averaged Navier-Stokes Model
Inviscid Flow, No Turbulence Models

RNG
Standard
Standard, realisable

Figure 9.5: *R–Desk* setup. Setting turbulence modelling and near—wall treatment (top), modelling approach options (left, bottom) and $k-\varepsilon$ model family variants (right, bottom).

, i.e. the Reynolds Averaged Navier–Stokes (RANS) equations will be solved. All approaches can be listed by left–clicking on the Turbulent Modelling Approach box, and they are shown in Figure 9.5 (left, bottom). Apart from RANS equations, the solution of laminar or inviscid flows (both do not require turbulence modelling) can be selected. The Turbulence Family box contains only $k-\varepsilon$ model family. Within each family, all available members are listed after left–click on the Turbulence Model ComboBox. The current models belonging to the $k-\varepsilon$ family are depicted in Figure 9.5 (right, bottom). These are the Standard $k-\varepsilon$ model, RNG based and Standard model with the realisable time scale bound. The default $k-\varepsilon$ model is the standard one. It remains to select the Near Wall Modelling method which can be either Low Re–number or Wall Functions approach. Three variants of wall functions are available, namely Scalable Wall Functions, Standard Wall Functions and Unified Wall Functions. The Standard wall functions are the default choice. The low Re–modelling option can be selected only for the Standard and Standard, realisable (i.e. TSB) models and it is always employed in conjunction with the Unified Wall Conditions .

10

MODELLING SINGLE–PHASE FLOWS

This chapter describes currently available physical models for the single–phase single– or multi–component:

- ◻ Fluid flow and associated
- ◻ Heat transfer and
- ◻ Mass transfer.

All the models are described by corresponding governing equations of *resolved* flow which ought to be closed by an Equation of state model equation. The flow–resolved equations are presented first, followed by the description of physical models and their selection by the user.

10.1 Governing Equations of Resolved Flow

Instantaneous equations for the mass, momentum and energy conservation describe both laminar and turbulent flows. The main strategies to deal with the problem of turbulence, namely DNS, RANS, LES and hybrid LES/RANS have been outlined in the introduction to turbulence modelling section. It can be noted that the above strategies result with an identical form of the governing equations. However, the meaning of flow variables is different: the variables represent instantaneous quantities in DNS, they are time or ensemble averaged quantities in RANS and filtered quantities in LES. As a common feature of these strategies is to *resolve* unsteady turbulent motion in time and space, the resulting conservation equations are known as the mean or resolved flow equations. The governing, single–phase resolved flow equations have the same form as the Reynolds–averaged equations. They read as follows:

- ◻ *Mass conservation*

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j}\left[\rho\left(U_j - U_{g,j}\right)\right] = s_m \tag{10.1}$$

  The source term s_m can be, for example, the mass source originated from from the dispersed phase (evaporation of liquid droplets) or any user defined source.

□ *Momentum conservation*

$$\frac{\partial}{\partial t}(\rho U_i)+\frac{\partial}{\partial x_j}\left[\rho U_i\left(U_j-U_{g,j}\right)\right]=-\frac{\partial p}{\partial x_i}+\rho f_i+\frac{\partial}{\partial x_j}\left(\tau_{ij}+\tau_{ij}^t\right) \tag{10.2}$$

In case of linear $k-\varepsilon$ models the turbulent stress tensors τ_{ij}^t is given by Equation (9.4). The sum of laminar and turbulent stresses can be then expressed as:

$$\tau_{ij}+\tau_{ij}^t=2\mu_{eff}\left(S_{ij}-\frac{1}{3}S_{nn}\delta_{ij}\right)-\frac{2}{3}\rho k\delta_{ij},\ \mu_{eff}=\mu+\mu_t \tag{10.3}$$

where the resolved mean strain tensor S_{ij} is given by Equation (9.5). The turbulent viscosity μ_t is predicted according to Equation (9.14). Molecular viscosity calculation options are discussed in relation to the setting of fluid phase properties, see also Table 8.1. Note that the pressure p in the above momentum equation represents the modified static pressure, defined by Equation (9.6). The body force (per mass unit mass) can include the effects of gravity (buoyancy) g_i, system rotation $f_{i,rot}$, porous media $f_{i,por}$ and user–defined forces $f_{i,usr}$:

$$f_i=g_i+f_{i,rot}+f_{i,por}+f_{i,usr} \tag{10.4}$$

Only the gravity body force is currently supported.

□ *Species mass fraction conservation*

$$\frac{\partial}{\partial t}(\rho c_i)+\frac{\partial}{\partial x_j}\left[\rho c_i\left(U_j-U_{g,j}\right)\right]=\frac{\partial}{\partial x_j}\left(J_{c_i,j}+J_{c_i,j}^t\right)+s_{c_i} \tag{10.5}$$

In the context of the $k-\varepsilon$ modelling the turbulent diffusion flux of species i, $J_{c_i,j}^t$, is given by Equation (9.8). Thus the sum of molecular and turbulent mass diffusion fluxes reads:

$$J_{c_i,j}+J_{c_i,j}^t=\mathscr{D}_{i,eff}\frac{\partial c_i}{\partial x_j},\ \mathscr{D}_{i,eff}=\mathscr{D}_i+\mathscr{D}_{i,t} \tag{10.6}$$

where the turbulent mass diffusion coefficient is defined by Equation (9.10). The section introducing mass transport highlights the role of the molecular mass diffusion coefficient $\mathscr{D}_i$. Its value should be specified for each species, see setting of fluid phase properties

□ *Energy conservation*

$$\underbrace{\frac{\partial}{\partial t}(\rho H)}_{\text{Unsteady Term}}+\underbrace{\frac{\partial}{\partial x_j}\left[\rho H\left(U_j-U_{g,j}\right)\right]}_{\text{Convection}}=\underbrace{\frac{\partial p}{\partial t}}_{\text{Pressure Change}}-\underbrace{\frac{\partial}{\partial x_j}\sum_{k=1}^{N_{sp}}h_k\left(J_{c_k,j}+J_{c_k,j}^t\right)}_{\text{Due to Species Diffusion}}$$
$$+\frac{\partial}{\partial x_j}\left[\underbrace{\left(q_j+q_j^t\right)}_{\text{Heat Fluxes}}+\underbrace{\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}}_{\text{Turb. Energy}}+\underbrace{U_i\left(\tau_{ij}+\tau_{ij}^t\right)}_{\text{Work of Stresses}}-\underbrace{pU_{g,j}}_{\text{Pressure Work}}\right]+\underbrace{\rho q_v}_{\text{Source}}+\underbrace{\rho f_i U_i}_{\text{Body Force Work}} \tag{10.7}$$

The above equation also outlines the physical meaning of various terms. Laminar and turbulent heat fluxes represent energy transport due to conduction. The work of viscous and turbulent stresses is also known as viscous dissipation. The source term might include the heat of chemical reaction or any user–defined source. In conjunction with eddy–viscosity $k-\varepsilon$ modelling the sum of laminar and turbulent heat fluxes is evaluated as:

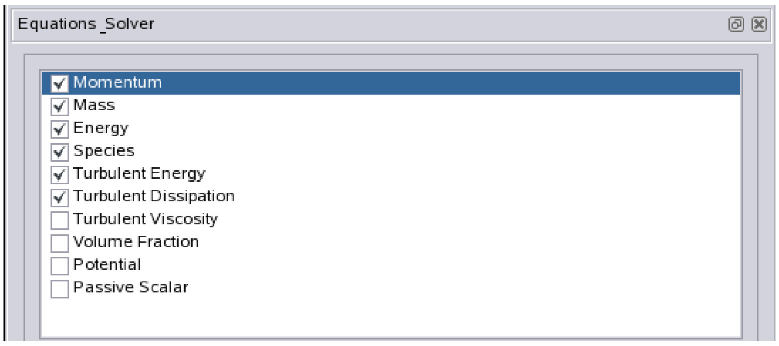
$$q_j + q_j^t = \lambda_{eff}\frac{\partial T}{\partial x_j}, \; \lambda_{eff} = \lambda + \lambda_t \tag{10.8}$$

The turbulent heat flux is modelled according to Equation (9.9) and the turbulent thermal conductivity λ_t is given by Equation (9.10). The molecular thermal conductivity, λ, is a fluid phase or species physical property which is specified as explained in the section about setting fluid phase properties. The effect of enthalpy transport due to species turbulent diffusion, represented by the correlation $\widehat{J_{c_k,j}h_k''}$ in Equation (6.31), can be modelled in a similar way as the effect due to species laminar diffusion. In Equation (10.7) these effects are taken into account for each species k. The total enthalpy H takes the usual form:

$$H = h + \frac{U_iU_i}{2} + k, \; h = \sum_k^{N_{sp}} c_k h_k \tag{10.9}$$

where h_k denotes specific thermal enthalpy of the k–th species and h is the thermal enthalpy of the fluid phase. For thermally perfect fluids the thermal enthalpy and internal energy are given by Equations (8.25) and (8.24), respectively. Their evaluation requires specific heat for which various calculation options have been presented in the section setting fluid phase properties.

Depending on the selected physical model, the *R–Desk* Equations_Solver panel will show which transport equations are to be solved, Figure 10.1. The momentum and mass equations are always


Equations_Solver

- ✓ Momentum
- ✓ Mass
- ✓ Energy
- ✓ Species
- ✓ Turbulent Energy
- ✓ Turbulent Dissipation
- ☐ Turbulent Viscosity
- ☐ Volume Fraction
- ☐ Potential
- ☐ Passive Scalar

Figure 10.1: *R–Desk* setup. List of equations solved for in VECTIS-MAX

activated as well as the energy equation. In case of incompressible fluid with constant physical properties, the energy equation is decoupled from others and the user might decide not to solve it. Apart from the transport equations for mass, momentum, species, energy and turbulence quantities, VECTIS-MAX provides a solution of equations for the volume fraction (in case of multi–phase modelling), velocity potential (in case of potential flow initialisation) and passive scalar.

10.2 Equation of State Models

The resolved flow equations need to be closed with an equation of state in order to determine density. Two models for the equation of state are available:

- ☐ Incompressible substance model: Density is either constant or depends on the temperature. The incompressible model is used for liquids, gases and solids.
- ☐ Ideal gas model: Density is a function of pressure and temperature and it is calculated from Equation(8.28):

$$\rho = \frac{p_{abs}}{RT} \tag{10.10}$$

with p_{abs} being the absolute pressure. The absolute pressure is defined as:

$$p_{abs} = p_s + p_{ref} \tag{10.11}$$

where p_s is the relative static pressure and p_{ref} denotes the reference pressure. The model is used only for gases. The density derivative with respect to the pressure under constant temperature, i.e. compressibility factor as defined by Equation (8.13), is also required when solving the pressure correction (continuity) equation, see Equation (14.58).

The equation of state model selection depends on the type of fluid phase (gas or liquid) and its compressibility options, see Figure 10.3. The incompressible model is used if a liquid or an incompressible or weakly compressible gas phase has been specified. A selection of the fully compressible sub–sonic or super–sonic gas implies the use of an ideal gas model. Considering density calculation options, all the options which define density with constant or temperature dependent (Boussinesq, exponential and polynomial) values can be used with the incompressible model. This also include the ideal gas option where now the constant, reference pressure value is used to calculate density:

$$\rho = \frac{p_{ref}}{RT} \tag{10.12}$$

If the equation of state is described by an ideal gas model then the ideal gas option should be selected as a density calculation option.

10.2.1 Reference and solver working pressure

The momentum conservation, see for example Equation (10.2), contains the pressure gradient $\partial p/\partial x_i$ whose accurate predictions are affected by the numerical round–off errors. In order to reduce the round– off errors, especially for incompressible and compressible low–Mach number flows, the concept of reference pressure, p_{ref}, is used to define the relative static pressure p_s:

$$p_s = p_{abs} - p_{ref} \tag{10.13}$$

which in turn defines the solver *computed or working* pressure. The solver working pressure, p, represents the pressure which is computed during solution of the resolved flow equations. Considering simulations involving the gravity (buoyancy) force it is given as:

$$p = p_s - \rho_{ref}\vec{g}\cdot\left(\vec{r}-\vec{r}_{ref}\right) + \left\{ \begin{array}{cl} \frac{2}{3}\mu\nabla\cdot\vec{U} & \rightarrow \text{laminar flows} \\ \frac{2}{3}\left(\mu_{eff}\nabla\cdot\vec{U}+\rho k\right) & \rightarrow \text{eddy–viscosity models} \end{array} \right. \tag{10.14}$$

The hydrostatic head in the above equation

$$\Delta p_h = \rho_{ref}\vec{g}\cdot\left(\vec{r}-\vec{r}_{ref}\right) \tag{10.15}$$

features the reference density ρ_{ref} and position vector of the reference altitude $\vec{r}_{ref}$ which is currently defined as $\vec{r}_{ref} = 0$. Thus the computed pressure in VECTIS-MAX is always relative to the reference pressure but it does not always represent the static pressure p_s. However, when computing density of an ideal gas, Equations (10.10, 10.11), it is assumed that $p_s \approx p$, i.e. the differences between the static and computed pressure are neglected.

The reference pressure itself and all input pressure values at various boundaries must be specified as absolute pressure values. For the gravity–driven flows the input values of the computed pressure p are expected and the post–processing results will be reported for the solver computed pressure.

The actual value of the reference pressure is not important in case of incompressible fluid model. However, if the ideal gas model is used the reference pressure value is important as the density depends on the absolute pressure, $p_{abs} = p + p_{ref}$. The reference pressure should be set to a value for which the computed pressure values are going to be small. In practice, it is often the average value of the specified inputs at pressure boundaries.

Closely related to the reference pressure is the *reference pressure location*. Its relevance depends on the existence of pressure boundaries in the considered fluid domain:

- If there are no pressure boundaries then the pressure value at one cell – the reference pressure location – must be fixed to the reference value in order to obtain the unique solution, Patankar [1980].
- If the pressure boundaries exist the reference pressure location is not relevant any more as the pressure level is set by specified boundary values.

Apart from the reference pressure, the specification of the reference temperature T_{ref} and density ρ_{ref} is important for simulations involving the gravity force. Setting of all reference variable values, including the reference pressure location is done via fluid domain panel, Figure (17.6).

10.3 Modelling Fluid Flow

Flow of any fluid phase can be modelled in accordance with various flow types. The next sections explain different flow models and how to select them.

10.3.1 Two– and Three–Dimensional Flows

Naturally, the fluid flows are three–dimensional. If the geometry, boundary and initial conditions lead to the numerical solution where flow variables changes in one fixed direction are not significant such a flow can be considered as two–dimensional. VECTIS-MAX supports planar flows taking place in the $x-y$ coordinate plane. Mathematically, the flow variables are functions of x, y coordinates and time t, $\phi = f(x,y,t)$. The velocity component in the z–direction is zero and it is not solved for. The selection of two– or three–dimensional simulation is done by checking one of Dimensionality RadioButtons in the Discretise panel, Figure 10.2.

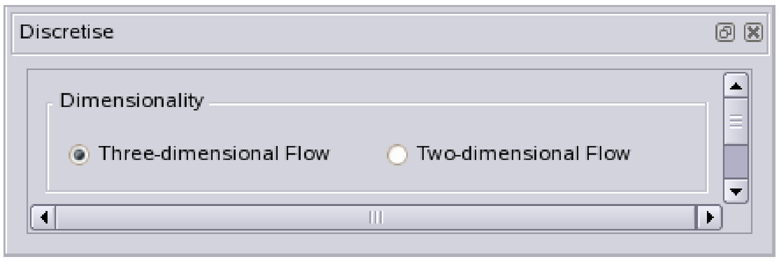


Figure 10.2: *R–Desk* setup. Selecting dimensionality of the simulation in Discretise panel.

This panel is open from the Solver Setup Tree under any Fluid Domain or Solid Domain node by left–clicking on the Discretise sub-node. Before selecting the two–dimensional simulation the user should ensure that the geometry is nominally two–dimensional, i.e. that

- two parallel and plane symmetry boundaries are placed in the z–direction (other boundary types are not allowed in this direction) and
- there is only a one layer of cells between symmetry planes in the z–direction. This might be difficult to achieve with VECTIS-MAX mesher using various levels of refinement. A refinement level (depth) of 0 does not do any refinement and with this option the mesh should have a one layer of cells in the z–direction.

10.3.2 Single and multi–component phase

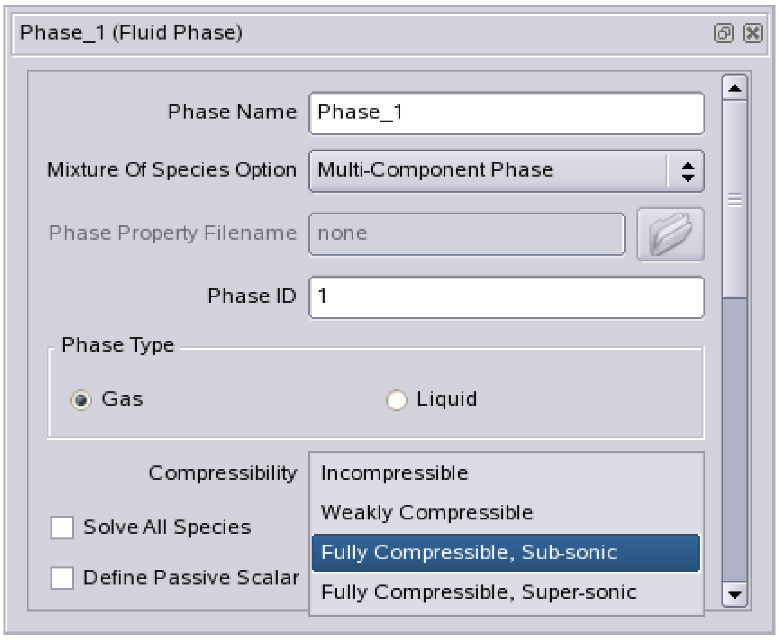
For either single–phase or multi–phase fluid flow the fluid phase can be selected as a single–component or multi–component, see Modelling Continua and Their Properties. The selection has been explained in the section dealing with Setting a phase and its properties with the help of Figure 8.3. Figure 10.3 shows the selection of multi–component phase under the Mixture of Species Option . If a multi–component phase is selected then the Solver Setup Tree will initially show two Fluid Species nodes.

10.3.3 Incompressible and compressible flow

The fluid compressibility is discussed in the section Equation of state & thermodynamic properties – its mathematical definition is given by Equation (8.13). The above Figure 10.3 also depicts the list of Compressibility options when the Gas is chosen as the Phase Type .

If the fluid phase is liquid then the density changes caused by pressure changes in Equation(8.13) are negligible and the fluid is considered as incompressible.

The selection of the phase type (gas or liquid) and a compressibility option determines the modelling of equation of state.


Phase_1 (Fluid Phase)

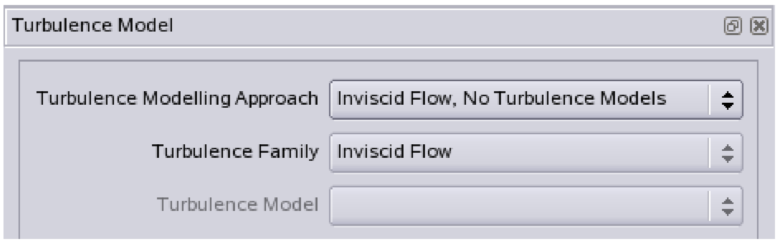
Phase Name: Phase_1
Mixture Of Species Option: Multi-Component Phase
Phase Property Filename: none
Phase ID: 1
Phase Type: [✓ Gas] Liquid
Compressibility: Incompressible / Weakly Compressible / [✓ Fully Compressible, Sub-sonic] / Fully Compressible, Super-sonic
Solve All Species
Define Passive Scalar

Figure 10.3: *R–Desk* setup. Setting a type of a fluid phase and its compressibility in Fluid Phase panel.

10.3.4 Inviscid and viscous regime

Fluid flows in nature are always *viscous* and described by the Navier–Stokes equations. For high–Reynolds number compressible flows *inviscid* flow model is sometimes used where the viscous and turbulent stress tensors in Equation (10.2)are neglected. The resulting momentum equations are then called *Euler equations*. The inviscid fluid can not stick to walls and therefore the slip wall boundary conditions are used.

The inviscid flow model is selected in the Turbulence Model panel after displaying with left–click the Turbulence Modelling Approach list box and selecting the Inviscid Flow option, Figure 10.4.


Turbulence Model

Turbulence Modelling Approach: Inviscid Flow, No Turbulence Models
Turbulence Family: Inviscid Flow
Turbulence Model:

Figure 10.4: *R–Desk* setup. Selecting inviscid flow regime in the Turbulence Model panel.

10.3.5 Potential flow model

An additional condition can be imposed on the inviscid flow: the irrotational velocity field $\nabla \times \vec{U} = 0$. This leads to the simplest flow model – potential flow. The velocity field can be then described by a scalar potential function – *a velocity potential* – Φ as

$$\vec{U} = \nabla\Phi \text{ or } U_i = \frac{\partial\Phi}{\partial x_i} \tag{10.16}$$

The equation for the velocity potential is obtained from the continuity equation as:

$$\frac{\partial\rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho\frac{\partial\Phi}{\partial x_j}\right) = 0 \tag{10.17}$$

Considering steady flow, and accounting for the isentropic conditions (see ideal gas model, Equation (8.31)) and that integration of the momentum gives the Bernoulli equation, the relation between density and the velocity potential can be established as follows:

$$\rho = \rho_{tot}\left(1 - \frac{(\nabla\Phi)^2}{2H_{tot}}\right)^{1/(\kappa-1)} \tag{10.18}$$

where ρ_{tot} and H_{tot} are the stagnation density and stagnation enthalpy, respectively. They are constant throughout the potential field.

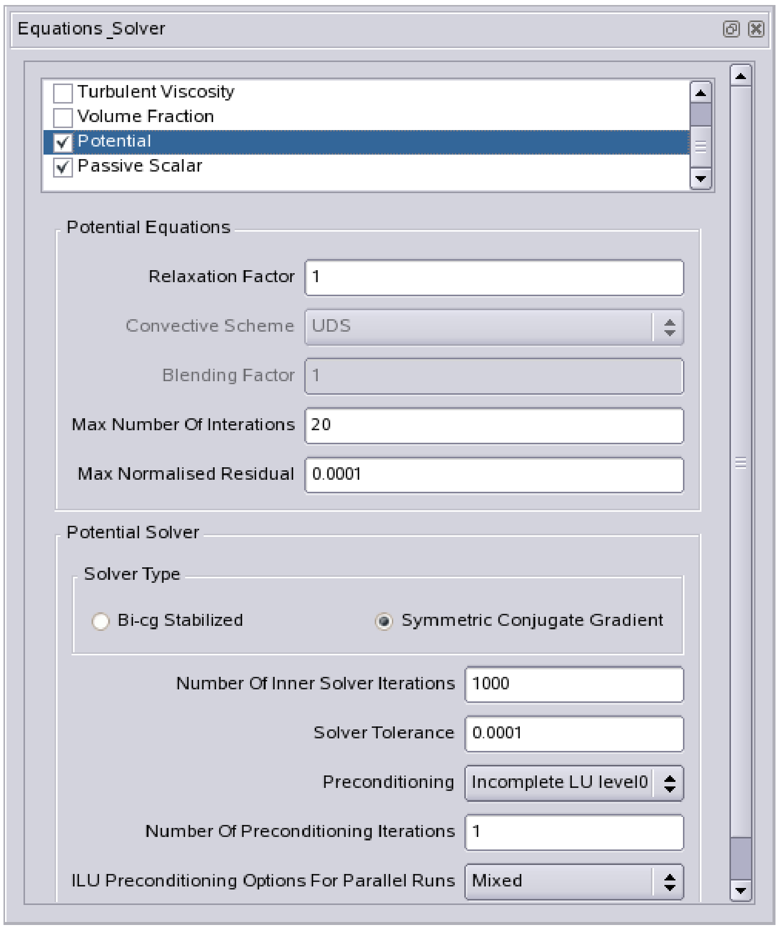
Clearly, potential flows are described by the diffusion type equation, where density plays a role of the diffusion coefficient which, in turn, depends on the magnitude of velocity potential. For the incompressible flows density is constant and Equation (10.17) becomes the Laplace equation.

VECTIS-MAX provides a solution of the steady potential flow as a part of the flow initialisation. The selection of potential flow initialisation is described in the Rdesk -fluid domain setup. If the potential flow initialisation is enabled the Potential equation (solver) in the Equations_Solver will be checked and the default control variables displayed as Figure 10.5 illustrates.

The Under–Relaxation Factor is used to under-relax density during an iterative solution of the velocity potential. The default value of unity should be always used for incompressible and subsonic flows. Its value might be reduced to $0.8 - 0.9$ if there are convergence problems in case of high Mach–number flows. Since the potential flow field is just used as an initial field the user is encouraged to experiment with the smaller Max Number of Iterations (default 20); sometimes two iterations can deliver the “good” initial velocity field. However, the Max Normalised Residuals should not be increased above 10^{-3}. Also, the Solver Tolerance of the linear equation solver should be below 10^{-3}.

10.3.6 Laminar and turbulent regime

Viscous flows are either *laminar* or *transitional* or *turbulent*. Basic physical aspects of turbulent flows are introduced in the section devoted to turbulence modelling. Laminar regime characterises the low Reynolds–number flows which are well–ordered and free of macroscopic random disturbances. Transitional flow regime encompasses the transition from a laminar to turbulent flow. The VECTIS-MAX can not predict *turbulence transition* and the available turbulence models are applicable only to fully turbulent flows. If it is a priory known that a flow regime is laminar this flow model is selected in the Turbulence Model panel in a similar way as the inviscid model, Figure 10.6.



Equations _Solver

- [ ] Turbulent Viscosity
- [ ] Volume Fraction
- [x] Potential
- [x] Passive Scalar

Potential Equations

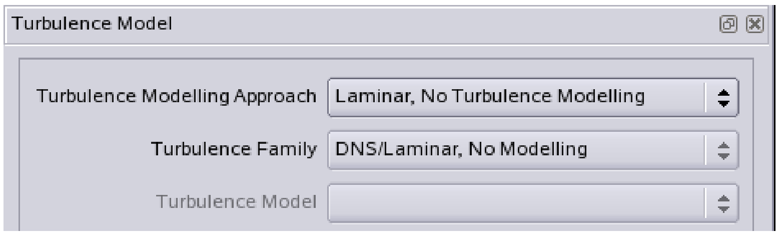
Relaxation Factor	1
Convective Scheme	UDS
Blending Factor	1
Max Number Of Interations	20
Max Normalised Residual	0.0001

Potential Solver

Solver Type: Bi-cg Stabilized [✓ Symmetric Conjugate Gradient]

Number Of Inner Solver Iterations	1000
Solver Tolerance	0.0001
Preconditioning	Incomplete LU level0
Number Of Preconditioning Iterations	1
ILU Preconditioning Options For Parallel Runs	Mixed

Figure 10.5: *R–Desk* setup. Setting the solution of the velocity potential in Equations_Solver panel.



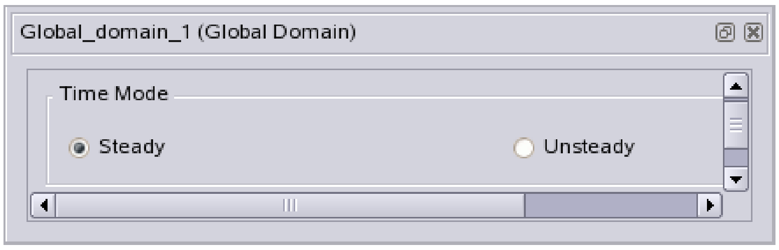
Turbulence Model

Turbulence Modelling Approach	Laminar, No Turbulence Modelling
Turbulence Family	DNS/Laminar, No Modelling
Turbulence Model	

Figure 10.6: *R–Desk* setup. Selecting laminar flow regime in the Turbulence Model panel.

10.3.7 Steady and unsteady flows

In principle, fluid flow characteristics are always time–dependent, i.e. the flows in nature are unsteady (transient). Many flows , however, can be defined as steady in the statistical sense, meaning that flow variables do not change significantly with time. A selection of the steady or unsteady simulation is performed within Global Domain panel where the Steady or Unsteady time mode can be specified, Figure 10.7.



Time Mode: [✓ Steady] Unsteady

Figure 10.7: *R–Desk* setup. Setting a steady or unsteady simulation in the Global Domain panel.

To perform unsteady simulations real time information are required such as the time step size, a number of time steps and in some situations the time dependent boundary conditions. The discretisation of the transient term, Figure 14.2, describes setting of steady and unsteady simulations.

If a steady state calculation converges it is reasonable to conclude that the simulated flow is steady. The same conclusion can be made after performing the unsteady simulation for which the time histories of monitoring flow variables indicate the steady state solution. The best way to confirm a steady state solution is to work with the optimal values of the time–step size and under–relaxation factors so that the solution converges within each time step after just one (outer) iteration per step.

Many flows are inherently unsteady, for example vortex shedding flows or buoyancy–driven flows, and they often show periodic behaviour. If such unsteady flows are run in steady mode their solution will not converge. However, if the simulated flow is physically steady, the transient behaviour indicated by the non-convergent solution in steady mode can be falsified by numerical effects. In such situations it is advisable to perform an unsteady simulation.

In case of unsteady simulations the physical time–step size is governed by the physics of the flow being simulated. For smaller time–step sizes more time steps are required but with fewer iterations per time step. Obviously there is a set of the optimal values for the time–step size, under–relaxation factors and for a number of iteration per time step. The user is advised to start with a small number of iteration per time step (5-10).

10.3.8 Gravity (buoyancy) –driven flows

Various body forces (per mass unit) are given in Equation (10.4) and these forces appear as volume sources in the momentum equation Equation (10.2). Currently, VECTIS-MAX provides modelling of the gravity force effects.

The gravity force per unit volume:

$$\vec{F}_g = \rho \vec{g} \tag{10.19}$$

always plays an important role in free surface and natural convection flows. The latter are flows driven by *thermal or concentration buoyancy* where gravity acts on the variable density field. In mixed convection flows the gravity force should not be neglected if the ratio of Grashof (Gr) and Reynolds (Re) numbers:

$$\frac{Gr}{Re^2} = \frac{|\vec{g}|\beta \Delta T_{ref} L_{ref}}{U_{ref}} \tag{10.20}$$

is close to or above one. In the above equation β represents the coefficient of volumetric expansion and ΔT_{ref}, L_{ref} and U_{ref} are reference temperature difference, length and velocity, respectively. Another dimensionless number, *Rayleigh number* (Ra):

$$Ra = GrPr = \frac{|\vec{g}|\beta \Delta T_{ref} L_{ref}^3 \rho^2 c_p}{\mu \lambda} \tag{10.21}$$

indicates the strength of the buoyancy induced flow in pure natural convection. The flow is considered laminar for $Ra < 10^8$ and the transition to turbulence happens over the range $10^8 < Ra < 10^{10}$.

The gravity force is conveniently cast in terms of constant (reference) and buoyancy source terms:

$$\vec{F}_g = \rho_{ref}\vec{g} + \left(\rho - \rho_{ref}\right)\vec{g} \tag{10.22}$$

The first term is then included in the solver working pressure, Equation (10.14, as:

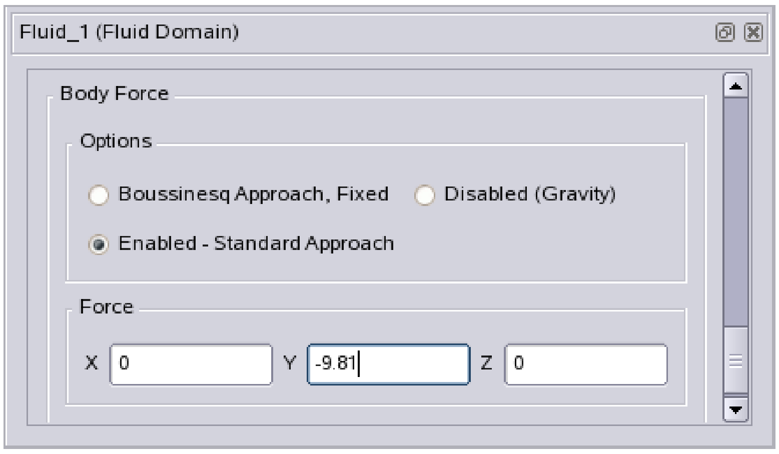
$$p = p_s - \rho_{ref}\vec{g} \cdot \left(\vec{r} - \vec{r}_{ref}\right) + \dots \tag{10.23}$$

In case of variable density flows, where density is a function of temperature, the buoyancy source $\left(\rho - \rho_{ref}\right)\vec{g}$ does not require any modelling (standard approach). If the constant density is going to be used for the buoyancy–driven flow throughout all equations then the Boussinesq model is used. Expressing density according to the Boussinesq approximation, Equation (8.27), the buoyancy source becomes:

$$\left(\rho - \rho_{ref}\right)\vec{g} = -\rho_{ref}\beta\left(T - T_{ref}\right)\vec{g} \tag{10.24}$$

The Boussinesq model should not be used for large density variations, i.e. it is applicable for $\beta\left(T - T_{ref}\right) << 1$.

The buoyancy model is selected in the Body Force sub–panel, which is a part of the Fluid Domain panel. As Figure 10.8 shows, by selecting the body force option via three RadioButtons the user can either disable gravity (default option) or enable the Boussinesq model for constant density flows or enable the standard approach for variable density flows. If the gravity is not disabled then the components of the Force per unit mass in x, y, z directions must be specified.


Fluid_1 (Fluid Domain)

Body Force

Options

( ) Boussinesq Approach, Fixed ( ) Disabled (Gravity)

(●) Enabled - Standard Approach

Force

X 0 Y -9.81 Z 0

Figure 10.8: *R–Desk* setup. Setting body force (buoyance) fluid model in Fluid Domain panel.

In addition, it is important to specify correctly the reference density $\rho_r ef$ and in case of the Boussinesq model the reference temperature T_{ref} and the coefficient of volumetric expansion β. Setting of these reference values is dealt with in the section which explains the fluid domain panel, Figure (17.6).

10.4 Modelling Heat Transfer

Three modes of heat transfer: conduction, convection and thermal radiation can occur within fluid and/or solid continuum. This section considers modelling of conductive and convective heat transfer in

- □ fluid domains,
- □ solid domains and
- □ conjugate heat transfer.

In each case a variation of the general energy equation will be solved.

10.4.1 Heat transfer in fluids

All convection modes – forced, natural and mixed – can be simulated by solving momentum and energy equations in fluids. Natural convection is associated with the buoyancy–driven flows.

The energy equation introduced earlier represents the conservation of total enthalpy. VECTIS-MAX solves the total enthalpy equation for the compressible flows (gases) while the total energy equation is solved for the incompressible flows. As the total energy can be expressed as:

$$E = e + \frac{U_i U_i}{2} + k = h - \frac{p}{\rho} + \frac{U_i U_i}{2} + k = H - \frac{p}{\rho} \tag{10.25}$$

its conservation equation can be obtained from the total enthalpy equation as:

$$\begin{aligned}
&\underbrace{\frac{\partial}{\partial t}(\rho E)}_{\text{Unsteady Term}} + \underbrace{\frac{\partial}{\partial x_j}\left[\rho E\left(U_j - U_{g,j}\right)\right]}_{\text{Convection}} = \underbrace{-\frac{\partial}{\partial x_j}\sum_{k=1}^{N_{sp}} h_k\left(J_{c_k,j} + J^t_{c_k,j}\right)}_{\text{Due to Species Diffusion}} \\
&+\frac{\partial}{\partial x_j}\left[\underbrace{\left(q_j + q^t_j\right)}_{\text{Heat Fluxes}} + \underbrace{\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}}_{\text{Turb. Energy}} + \underbrace{U_i\left(\tau_{ij} + \tau^t_{ij}\right)}_{\text{Work of Stresses}} - \underbrace{pU_j}_{\text{Pressure Work}}\right] + \underbrace{\rho q_v}_{\text{Source}} + \underbrace{\rho f_i U_i}_{\text{Body Force Work}}
\end{aligned} \tag{10.26}$$

Based on the selection of the fluid compressibility the total energy equation will be solved for incompressible flows and total enthalpy equation for all compressible flows.

An important issue when solving either the total enthalpy or energy equation is the design of a conservative discretisation scheme. As Equation (10.26) shows, the convective and unsteady terms are expressed in terms of total enthalpy or energy while the diffusion term is naturally defined in terms of the temperature gradient. It is beneficial, especially in case of conjugate heat transfer modelling, to discretise and solve the energy equation in terms of temperature as a primary variable. For this, an efficient numerical technique is devised and described in the section presenting the discretisation of the energy equation. If the energy equation is solved in terms of total enthalpy or energy the temperature field is extracted according to the definition of total enthalpy/energy, Equation (10.9 or 10.25) During iterative solution the new temperature is under–relaxed by using the pressure under–relaxation.

Thus when solving the energy equation in fluid domains the user can specify the solution in terms of either Enthalpy or Temperature by activating the corresponding RadioBoxes in the Equations_Solver panel shown in Figure 10.9. The default Under–relaxation Factor , Convective

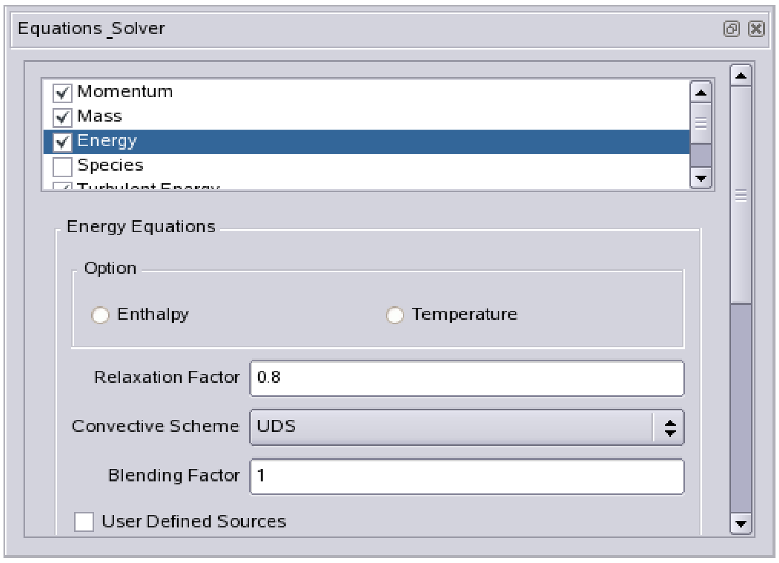


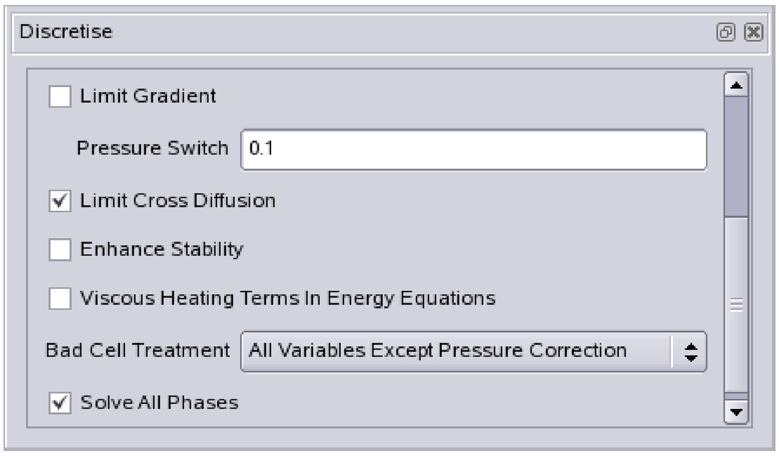
Figure 10.9: *R–Desk* setup. Setting solution of the energy equation in the Equations_Solver panel.

Scheme and Blending Factor for convective schemes are also shown. The selection of a convective scheme is discussed in the section Interpolative schemes for cell–face values, see Figure 14.5. With reference to the same figure, the setting of parameters which control the linear equation solver(s) can be found in the section Linear solvers. If there is a User Defined Source it should be activated in this panel. Note that higher values of the under–relaxation factor should be tried (0.9) for the solution in terms of temperature.

The heat flux due to species diffusion in Equations (10.26, 10.7) is not currently implemented. Some other terms on the right–hand side of the above equations can be neglected in some flow situations. Typically, pressure work and the work of stresses (i.e. viscous dissipation or heating) term are often negligible in incompressible flows. However, VECTIS-MAX always account for all terms except for the viscous heating term. This term should be included when the Brinkman number (Br):

$$Br = \frac{\mu U_{ref}^2}{\lambda \Delta T_{ref}} \qquad (10.27)$$

is close or above one.


Discretise

- [ ] Limit Gradient
- Pressure Switch: 0.1
- [x] Limit Cross Diffusion
- [ ] Enhance Stability
- [ ] Viscous Heating Terms In Energy Equations
- Bad Cell Treatment: All Variables Except Pressure Correction
- [x] Solve All Phases

Figure 10.10: *R–Desk* setup. Inclusion of viscous heating in the solution of the energy equation

The inclusion of the viscous heating term is done via Discretise panel, Figure 10.10, where the Viscous Heating Term in Energy Equation can be ticked on.

10.4.2 Heat transfer in solids

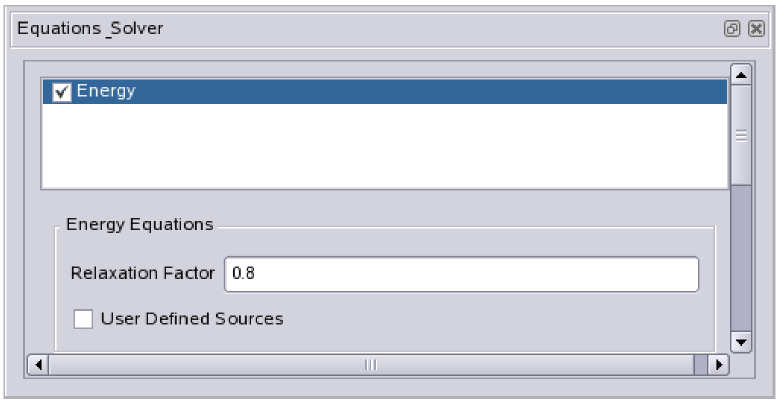
In solid regions, the heat conduction is the only mode of heat transfer and the total energy equation (10.26) is reduced to the simplified, internal energy form:

$$\frac{\partial \rho e}{\partial t} = \frac{\partial}{\partial x_j}\left(\lambda \frac{\partial T}{\partial x_j}\right) + \rho q_v \tag{10.28}$$

where internal energy of a solid is defined as $e = \int_{T_{ref}}^{T} c_v T$. Currently, the isotropic thermal conductivity λ must be used. However, the solid solution domain can have an arbitrary number of solid materials, see modelling of spatial domains and also the multi–domain example in Figure 7.1. In case of multi–material domain, the grids along material interfaces must be conformal. To achieve this, the arbitrary grid interface tool is available. The solid energy equation is always solved in terms of temperature as a primary variable.

To set up the solution of the energy equation in solids, it should be activated in the solid Equations_Solver panel, see Figure 10.11. In addition, User Defined Sources can be included.

In case of constant solid properties, the finite volume discretisation leads to a system of linear equations with constant coefficients which should converge in one outer iteration if the grid non-orthogonality is absent. For such cases, attempts should be made to use optimal, i.e. maximum under-relaxation factors - one or very close to one. In the presence of poor quality cells the user should try to run cases without the poor quality cell treatment.


Equations_Solver

☑ Energy

Energy Equations

Relaxation Factor 0.8

☐ User Defined Sources

Figure 10.11: *R–Desk* setup. Setting solution of the energy equation for solid domains in the Equations_Solver panel.

10.4.3 Conjugate heat transfer

Conjugate Heat Transfer (CHT) describes coupled heat transfer through adjacent fluid and solid material domains. There are numerous engineering applications where detailed CHT analyses are required. Typical automotive applications such as cooling of cylinder heads and engine blocks as well as under–hood aero–thermal management illustrate well the challenges associated with modelling of CHT in complex geometries.

The total enthalpy equation (10.7) or total energy equation (10.26) is solved in terms of temperature in a fully implicit manner over the entire global domain, see also the multi–domain example in Figure 7.1. The conformal meshes at material interfaces should be provided with the help of arbitrary grid interface tool. The discretisation of diffusion fluxes at interfaces is dealt with in the section Discretisation of the energy equation.

There are no special inputs related to the setting of CHT problems. Based on the supplied multi–domain structure, i.e. when there is at least one fluid and one solid domain, the solver will activate the CHT solution.

10.5 Modelling Mass Transfer

Mass transfer is analogous to heat transfer. As heat is transferred from regions of high temperature to regions of low temperature, the mass of one species travels from regions of high concentration to regions of low concentration. By its nature, mass transfer occurs within multi–component phase (continuum) which is defined as mixture of species. The molecular mass transfer of species by diffusion, including the definition of the species concentration or mass fraction c_i, is outlined in Section Mass transport and mass diffusion coefficients. All mechanisms of mass transfer – molecular diffusion, convection and turbulent transport – are described by the species modelling equation. Thermo–physical properties featuring in the mass, momentum and energy equations are properties of mixture of species. They depend on the species mass fractions and their prediction is explained in Section Properties of Multicomponent Phase

Note that the transport of energy due to species diffusion in Equations (10.7, 10.26):

$$\frac{\partial}{\partial x_j}\sum_{k=1}^{N_{sp}} h_k\left(J_{c_k,j}+J^t_{c_k,j}\right)$$

is currently neglected. This is justified when the Lewis number (Le):

$$Le_k = \frac{\lambda}{\rho c_p \mathscr{D}_k} \tag{10.29}$$

for each species k is much greater than unity.

The mass transfer is an important ingredient of reacting flows (including combusting flows). The reacting flows are also described by the species modelling equation but are not yet supported.

Modelling mass transfer implies setting of the multi–component phase, see also Figure 8.3. If either multi–component or WAVE –based phase is selected this will enable the definition of constituent species (the WAVE –based phase has the pre–defined set of species) and their thermo–physical properties, see Figure 8.10 and Figure 8.11. The Species modelling equation in the Equations_Solver panel, Figure 10.1, will be also activated.

The computed values of mass fractions c_i have to satisfy the compatibility condition, $\sum_i^{N_{sp}} c_i = 1$. This condition can be enforced by either solving mass fractions for all species or solving for $N_{sp}-1$ species and calculate the mass fraction of the last species as

$$c_{N_{sp}} = 1 - \sum_{i=1}^{N_{sp}-1} c_i. \tag{10.30}$$

The Fluid Phase panel, shown in Figure 10.3, contains the Solve All Species CheckBox. By default the box is not checked, i.e. $N_{sp}-1$ species will be solved for.

10.5.1 Passive scalar

Transport of a passive scalar is governed by the same equation as standard species. As the attribute “passive” suggests, the thermo–physical properties of the parent fluid phase are not affected by the passive scalar properties, i.e. by their mass fractions. Section Passive scalar and its properties should be consulted about setting of a passive scalar.

11

MODELLING POROUS MEDIA

A porous medium is a three–dimensional region occupied by continuum which comprises both fluid material and fine–scale solid structure. The structure and its interstices (pores), through which fluid permeates, are usually too small to be resolved by computational mesh. Exhaust catalysts, packed columns, heat exchangers, filters, flow distributors and tube bundles are some of porous media application examples.

In terms of VECTIS-MAX multi–domain structure, a porous media region is associated with the porous fluid sub–domain; an example is shown in Figure 7.4. The appropriate mesh joining procedure should be used to obtain conformal meshes at interfaces between a porous sub-domain and its parent fluid domain. In some situations, the sub–domain geometry can be approximated by infinitely thin region, represented by cell-faces, i.e. by interface between cells rather than a cell region. This kind of simplification is described by so called *porous jump conditions* and it is not yet supported.

A porous medium is typically characterised by directional pressure drop which is either linear or non-linear function of the local velocity. For each porous media type, the pressure drop is determined by the corresponding flow–resistance model. The following porous media types (resistance models) are available:

- □ General porous media: isotropic or orthotropic media with the flow resistance defined in terms of viscous and inertial resistance tensors (Forchheimer's model).
- □ Catalytic converters: orthotropic media with Darcy's viscous resistance model, i.e. the pressure drop depends linearly on the local velocity.
- □ General orthotropic media: inertial resistance model, where the pressure drop is a quadratic function of the local velocity.
- □ Radiators: orthotropic media with both viscous and inertial resistance models included.
- □ General heat exchangers: the flow resistance as for the general porous media.

The next section presents porous media modelling theoretical background, including the governing equations. Then currently supported porous model types are described as well as the required user inputs for each model type.

11.1 Theoretical Background

A fluid flow through a permeable structure is described by instantaneous Navier–Stokes equations. In practice, ensemble averaging, resulting with the Reynolds–averaged equations (RANS), is employed to address the turbulence problem. However, the existence of the fine porous structure makes even RANS simulations computationally impractical and further volume averaging at an affordable macroscopic scale (which accounts for the porous structure) is required.

11.1.1 Volume averaging procedure

In the Volume Averaging Theory (VAT), see for example Whitaker [1999]; Slattery [1999], grid elements (control volumes) contain both fluid and solid phases, i.e. small–scale solid structures and fluid pores are not resolved with a numerical grid. Instead, the numerical grid, often associated with a Representative Elementary Volume (REV), should be fine enough to resolve *macroscopic* flow and temperature fields. One can expect that governing equations, describing these macroscopic fields, account for interactions between fluid and solid parts within a REV.

Considering a general variable ϕ, the *superficial* volume average $\langle\phi\rangle^s$ and *intrinsic* volume average $\langle\phi\rangle$ operators can be introduced as:

$$\langle\phi\rangle^s(x_k,t) = \frac{1}{V}\oint_V \phi(\Delta x_k,t)\,\gamma_f(\Delta x_k,t)dV \tag{11.1}$$

$$\langle\phi\rangle(x_k,t) = \frac{1}{V_f}\oint_V \phi(\Delta x_k,t)\,\gamma_f(\Delta x_k,t)dV \tag{11.2}$$

where an averaging volume $V = V_f + V_s$ consists of the fluid V_f and solid V_s parts. The position vector Δx_k of any point inside the volume is defined relative to the centre of the volume while the vector x_k denotes the position vector of the volume centre. A fluid phase distribution function γ_f, defined as:

$$\gamma_f = \begin{cases} 1\,, & \text{if } \Delta x_k \in V_f \\ 0\,, & \text{if } \Delta x_k \in V_s \end{cases} \tag{11.3}$$

identifies the fluid volume. The superficial and intrinsic averages are related through the volume porosity γ:

$$\langle\phi\rangle^s = \gamma\langle\phi\rangle\ , \gamma = \langle\gamma_f\rangle^s = \frac{V_f}{V} \tag{11.4}$$

Similarly to the Reynolds and Favre decomposition, Equations (6.17) and (6.22) respectively, the variable ϕ can be split into its intrinsic volume average $\langle\phi\rangle$ and the spatial deviation of ϕ from the intrinsic average, $\breve{\phi}$:

$$\phi = \langle\phi\rangle + \breve{\phi}\ ,\ \text{with } \langle\breve{\phi}\rangle = 0 \tag{11.5}$$

The usual rules for a sum and a product of two variables apply:

$$\langle\phi+\psi\rangle = \langle\phi\rangle + \langle\psi\rangle\ , \langle\phi\psi\rangle = \langle\phi\rangle\langle\psi\rangle + \langle\breve{\phi}\breve{\psi}\rangle\ , \langle\breve{\phi}\langle\phi\rangle\rangle = 0 \tag{11.6}$$

In order to derive the volume averaged RANS equations, it is necessary to recall two important theorems which relate volume averages of derivatives to derivatives of averages. The local averaging theorem, see for example Whitaker [1999], is given as:

$$\langle\frac{\partial\phi}{\partial x_i}\rangle^s = \frac{\partial\gamma\langle\phi\rangle}{\partial x_i} + \frac{1}{V}\int_{A_i}\phi n_i dA \text{ or } \langle\frac{\partial\phi}{\partial x_i}\rangle = \frac{1}{\gamma}\left[\frac{\partial\gamma\langle\phi\rangle}{\partial x_i} + \frac{1}{V}\int_{A_i}\phi n_i dA\right] \tag{11.7}$$

where A_i represents the interfacial area between fluid and solid inside a REV and n_i is the unit outward vector normal to A_i. The above surface integral introduces into volume averaged equations additional *dispersion* terms which describe microscopic interactions between fluid and solid phases. The transport theorem relates temporal derivatives:

$$\langle\frac{\partial\phi}{\partial t}\rangle^s = \frac{\partial\gamma\langle\phi\rangle}{\partial t} - \frac{1}{V}\int_{A_i}\phi n_i U_i dA \text{ or } \langle\frac{\partial\phi}{\partial t}\rangle = \frac{1}{\gamma}\left[\frac{\partial\gamma\langle\phi\rangle}{\partial t} - \frac{1}{V}\int_{A_i}\phi n_i U_i dA\right] \tag{11.8}$$

where U_i is the fluid velocity vector at fluid–solid interface. For non-moving interfaces the surface integral vanishes.

11.1.2 Double–decomposition concept

Note that volume averaging can be applied to the instantaneous quantity $\phi \equiv \widehat{\phi} = \overline{\phi} + \phi' = \widetilde{\phi} + \phi''$ or to the ensemble $\phi \equiv \overline{\phi}$ or Favre $\phi \equiv \widetilde{\phi}$ averaged quantity. Also, the ensemble/Favre averaging can be applied to the volume averaged quantity. In this way, one can arrive at the double decomposition concept for an instantaneous variable $\widehat{\phi}$ as shown by de Lemos [2005]:

$$\widehat{\phi} = \langle\phi\rangle + \breve{\phi} = \overline{\langle\phi\rangle} + \langle\phi\rangle' + \overline{\breve{\phi}} + (\breve{\phi})' \tag{11.9}$$

$$\widehat{\phi} = \overline{\phi} + \phi' = \langle\overline{\phi}\rangle + \breve{\overline{\phi}} + \langle\phi'\rangle + (\breve{\phi'}) \tag{11.10}$$

In the case of the rigid porous medium the above averaging operators commute:

$$\overline{\langle\phi\rangle} = \langle\overline{\phi}\rangle\ , \langle\phi\rangle' = \langle\phi'\rangle\ , \overline{\breve{\phi}} = \breve{\overline{\phi}}\ , (\breve{\phi})' = (\breve{\phi'}) \tag{11.11}$$

11.1.3 Governing equations

Macroscopic equations describing turbulent flow in a porous medium can be derived in two ways: either applying ensemble/Favre averaging and then volume averaging or using a reverse order of these averaging procedure. As long as the averaging operators commute the final form of the mass, momentum and energy equations will not be affected by order of averaging.

Thus we can start with the governing equations of resolved flow, Eqs. (10.1), (10.2), (10.5) and (10.7), and perform spatial averaging over a representative control volume. Considering a rigid porous medium (non-moving control volumes), with the no–slip condition on the fluid–solid interface, and denoting double–averaged variables without double-average notations ($\phi \equiv \langle\overline{\phi}\rangle$, $\phi^s \equiv \langle\overline{\phi}\rangle^s$), conservation equations describing porous media can be derived as:

□ *Mass conservation*

$$\frac{\partial \gamma\rho}{\partial t} + \frac{\partial}{\partial x_j}\left(\gamma\rho U_j\right) = \gamma s_m - \frac{\partial}{\partial x_j}\left(\gamma\langle\breve{\overline{\rho}}\breve{\widetilde{U}}_j\rangle\right) \tag{11.12}$$

The last term in the above equation represents mass dispersion. This term is neglected as spatial fluctuations of the ensemble–mean density can be considered small in comparison to the volume–averaged ensemble–mean density, i.e. $\overline{\rho} = \langle\overline{\rho}\rangle + \breve{\overline{\rho}} \approx \langle\overline{\rho}\rangle$. Neglecting mass dispersion effectively eliminates a large number of density–related terms arising from the volume averaging procedure. Consequently, the volume–averaged ideal gas law will keep the original functional form:

$$p = \langle\overline{p}\rangle = R_g\langle\overline{\rho}\widetilde{T}\rangle = R_g\langle\overline{\rho}\rangle\langle\widetilde{T}\rangle + R_g\langle\breve{\overline{\rho}}\breve{\widetilde{T}}\rangle \approx R_g\langle\overline{\rho}\rangle\langle\widetilde{T}\rangle = R_g\rho T \tag{11.13}$$

□ *Momentum conservation*

$$\frac{\partial}{\partial t}\left(\gamma\rho U_i\right) + \frac{\partial}{\partial x_j}\left(\gamma\rho U_i U_j\right) = -\frac{\partial \gamma p}{\partial x_i} + \gamma\rho f_i + \frac{\partial}{\partial x_j}\left[\gamma\left(\tau_{ij} + \tau^t_{ij}\right)\right] + f_{p_i} \tag{11.14}$$

where f_{p_i} represents the resistance force per unit volume to flow in the porous medium. This force is given by

$$f_{p_i} = \frac{1}{V}\int_{A_i}\left(\overline{\tau}_{ij} - \overline{p}\delta_{ij}\right)n_j dA - \frac{\partial}{\partial x_j}\left(\gamma\overline{\rho}\langle\breve{\widetilde{U}}_i\breve{\widetilde{U}_j}\rangle\right) \tag{11.15}$$

The last term on the right–hand side of the above equation, often called *inertial dispersion*, is associated with spatial fluctuations of the ensemble–mean velocities.

For gravity–driven flows, the double–averaged $p = \langle\overline{p}\rangle$ and ensemble–mean $\overline{p}$ pressure in the above equations represent the modified pressure which includes the reference buoyancy force, see Equation (10.23)

The macroscopic intrinsic and superficial viscous stresses, τ_{ij} and τ^s_{ij}, are defined as:

$$\tau_{ij} = \langle\overline{\tau}\rangle_{ij} = 2\mu\left(S_{ij} - \frac{1}{3}S_{nn}\delta_{ij}\right), \tau^s_{ij} = \langle\overline{\tau}\rangle^s_{ij} = \gamma\tau_{ij} = 2\mu\left(S^s_{ij} - \frac{1}{3}S^s_{nn}\delta_{ij}\right) \tag{11.16}$$

where the macroscopic intrinsic and superficial strain tensors, S_{ij} and S^s_{ij}, respectively, read:

$$S^s_{ij} = \langle\overline{S}\rangle^s_{ij} = \frac{1}{2}\left(\frac{\partial\gamma U_i}{\partial x_j} + \frac{\partial\gamma U_j}{\partial x_i}\right), S_{ij} = \langle\overline{S}\rangle_{ij} = \frac{S^s_{ij}}{\gamma} \text{ and } S^s_{kk} = \frac{\partial\gamma U_k}{\partial x_k}, S_{kk} = \frac{S^s_{kk}}{\gamma} \tag{11.17}$$

In the case of linear $k-\varepsilon$ models, the turbulent stress tensors τ^t_{ij} is given by Equation (9.4). After volume averaging, its macroscopic counterpart becomes:

$$\tau^t_{ij} = \langle -\overline{\rho}\widetilde{U''_i U''_j}\rangle = 2\mu_{t\gamma}\left(S_{ij} - \frac{1}{3}S_{nn}\delta_{ij}\right) - \frac{2}{3}\rho k\delta_{ij} \tag{11.18}$$

where $\mu_{t\gamma}$ signifies the macroscopic turbulent viscosity. It can be defined similarly to the microscopic turbulent viscosity, see Equation (9.14), where now $k = \langle\widetilde{u''_i u''_i}\rangle = \langle\widetilde{k}\rangle$ and ε denote the macroscopic (volume–averaged) turbulent kinetic energy and its dissipation rate.

Considering the isotropic porous medium, closure for the porous resistance force, Equation (11.15), is commonly provided by Darcy–Forchheimer's model:

$$f_{p_i} = \gamma \left(\frac{\mu}{K} + \rho C_F |\vec{U}^s| \right) U_i^s \tag{11.19}$$

where K $[m^2]$ and C_F $[m^{-1}]$ are the permeability and Forchheimer's parameters, respectively, which in general depend on the volume porosity γ. The Forchheimer's parameter is often expressed as $C_F = C_F'/\sqrt{K}$, where C_F' is the Forchheimer's non-dimensional constant. Introducing the permeability and Forchheimer's tensors (K_{ij} and C_{ij}, respectively), the total porous resistance tensor $\mathscr{R}_{ij}$ can be defined in terms of the viscous (linear) $\mathscr{R}_{ij}^v$ and inertial (quadratic) $\mathscr{R}_{ij}^i$ parts as follows:

$$\mathscr{R}_{ij} = \mathscr{R}_{ij}^v + \mathscr{R}_{ij}^i |\vec{U}^s| \text{ with } \mathscr{R}_{ij}^v = \mu K_{ij}^{-1} \text{ and } \mathscr{R}_{ij}^i = \rho C_{ij} \tag{11.20}$$

This leads to the generalised Darcy–Forchheimer's model applicable to non–isotropic porous media:

$$f_{p_i} = \gamma \left(\mu K_{ij}^{-1} + \rho C_{ij} |\vec{U}^s| \right) U_j^s = \gamma \left(\mathscr{R}_{ij}^v + \mathscr{R}_{ij}^i |\vec{U}^s| \right) U_j^s \tag{11.21}$$

The viscous and inertial tensors accommodate various flow resistance models whose parameters are user–supplied. Their dimensions are $[kg/(m^3 s)]$ and $[kg/m^4]$, respectively. It can be noticed that the superficial volume–averaged velocity $U_i^s = \langle \widetilde{U}_i \rangle^s$ is used in the Darcy–Forchheimer's model.

□ *Species mass fraction conservation*

$$\frac{\partial}{\partial t}(\gamma \rho c_i) + \frac{\partial}{\partial x_j}(\gamma \rho c_i U_j) = \frac{\partial}{\partial x_j}\left[\gamma \left(J_{c_i,j} + J_{c_i,j}^t - \overline{\rho} \langle \breve{\widetilde{c}}_i \breve{\widetilde{U}}_j \rangle \right) \right] + \gamma s_{c_i} + \frac{1}{V}\int_{A_i} \overline{J}_{c_i,j} dA \tag{11.22}$$

where the intrinsic molecular diffusion flux $J_{c_i,j}$ is defined as:

$$J_{c_i,j} = \langle \overline{J}_{c_i,j} \rangle = \frac{1}{\gamma} \left(\mathscr{D}_i \frac{\partial (\gamma c_i)}{\partial x_j} + \frac{1}{V} \int_{A_i} \mathscr{D}_i \widetilde{c}_i n_j dA \right) \tag{11.23}$$

The last term in the above equation can be modelled in terms of the intrinsic concentration gradient:

$$\frac{1}{V} \int_{A_i} \mathscr{D}_i \widetilde{c}_i n_j dA = \mathscr{D}_{i,jk}^{tor} \frac{\partial (\gamma c_i)}{\partial x_k} \tag{11.24}$$

where $\mathscr{D}_{i,jk}^{tor}$ denotes the *tortuosity diffusion tensor*.

In the context of the $k-\varepsilon$ modelling, the turbulent diffusion flux of species i is given by Equation (9.8). Its volume–averaged counterpart is given as:

$$J_{c_i,j}^t = \langle -\overline{\rho} \widetilde{c_i'' u_j''} \rangle = \frac{1}{\gamma} \left(\mathscr{D}_{i,t} \frac{\partial (\gamma c_i)}{\partial x_j} \right) \tag{11.25}$$

where the turbulent mass diffusion coefficient $\mathscr{D}_{i,t}$ is defined by Equation (9.10). The concentration dispersion term $-\overline{\rho} \langle \breve{\widetilde{c}}_i \breve{\widetilde{U}}_j \rangle$ can be also expressed as a function of the intrinsic concentration gradient:

$$-\overline{\rho} \langle \breve{\widetilde{c}}_i \breve{\widetilde{U}}_j \rangle = \frac{1}{\gamma} \mathscr{D}_{i,jk}^{dis} \frac{\partial (\gamma c_i)}{\partial x_k} \tag{11.26}$$

by introducing the dispersive diffusion tensor $\mathscr{D}^{dis}_{i,jk}$. Finally, the last term in Equation (11.22) represents the mass flux at fluid–solid interface which is zero.

The above mass diffusion coefficients can be lumped together, defining an effective mass diffusion tensor:

$$\mathscr{D}^{e}_{i,jk} = (\mathscr{D}_i + \mathscr{D}_{i,t})\,\delta_{jk} + \mathscr{D}^{tor}_{i,jk} + \mathscr{D}^{dis}_{i,jk} \tag{11.27}$$

With this definition, Equation (11.22) becomes:

$$\frac{\partial}{\partial t}(\gamma\rho c_i) + \frac{\partial}{\partial x_j}(\gamma\rho c_i U_j) = \frac{\partial}{\partial x_j}\left[\mathscr{D}^{e}_{i,jk}\frac{\partial(\gamma c_i)}{\partial x_k}\right] + \gamma s_{c_i} \tag{11.28}$$

□ *Energy conservation*

Volume averaging of the simplified fluid phase energy Equation (10.7) (fluxes due to species diffusion omitted) gives:

$$\frac{\partial}{\partial t}(\gamma\rho H) + \frac{\partial}{\partial x_j}(\gamma\rho H U_j) = \frac{\partial\gamma p}{\partial t} + \frac{\partial}{\partial x_j}\left[\gamma\left(q_j + q^t_j - \overline{\rho}\langle\breve{\tilde{H}}\breve{U}_j\rangle + U_i\left(\tau_{ij} + \tau^t_{ij}\right)\right) + \frac{\mu_t}{\sigma_k}\frac{\partial\gamma k}{\partial x_j}\right]$$
$$+\frac{1}{V}\int_{A_i}(\lambda + \lambda_t)\frac{\partial\widetilde{T}}{\partial x_j}n_j dA + \gamma\rho q_v + \gamma\rho f_i U_i \tag{11.29}$$

The intrinsic laminar heat flux is defined as:

$$q_j = \langle\overline{q}_j\rangle = \frac{1}{\gamma}\left(\lambda\frac{\partial(\gamma T)}{\partial x_j} + \frac{1}{V}\int_{A_i}\lambda\widetilde{T}n_j dA\right) \tag{11.30}$$

The last term in the above equation is modelled with the help of *thermal tortuosity conductivity tensor* λ^{tor}_{jk}

$$\frac{1}{V}\int_{A_i}\lambda\widetilde{T}n_j dA = \lambda^{tor}_{jk}\frac{\partial(\gamma T)}{\partial x_k} \tag{11.31}$$

In conjunction with eddy–viscosity $k-\varepsilon$ modelling, the volume–averaged turbulent heat flux q^t_j is evaluated according to Equation (9.9):

$$q^t_j \approx \langle -\overline{\rho}\overline{c}_p\widetilde{T''u''_j}\rangle = \frac{1}{\gamma}\left(\lambda_t\frac{\partial(\gamma T)}{\partial x_j}\right) \tag{11.32}$$

where the turbulent thermal conductivity λ_t is given by Equation (9.10). The *thermal dispersion* term, $-\overline{\rho}\langle\breve{\tilde{H}}\breve{U}_j\rangle$, is usually modelled as a function of the intrinsic temperature gradient:

$$-\overline{\rho}\langle\breve{\tilde{H}}\breve{U}_j\rangle \approx -\overline{\rho}\overline{c}_p\langle\breve{\tilde{T}}\breve{U}_j\rangle = \frac{1}{\gamma}\lambda^{dis}_{jk}\frac{\partial(\gamma T)}{\partial x_k} \tag{11.33}$$

by introducing the thermal dispersion conductivity tensor λ^{dis}_{jk}.

One can define an effective fluid conductivity tensor

$$\lambda^{e}_{jk} = (\lambda + \lambda_t)\,\delta_{jk} + \lambda^{tor}_{jk} + \lambda^{dis}_{jk} \tag{11.34}$$

and calculate the sum of various heat fluxes in terms of the effective heat flux:

$$q^e_j = q_j + q^t_j - -\overline{\rho}\langle\breve{\tilde{T}}\breve{U}_j\rangle = \frac{1}{\gamma}\lambda^{e}_{jk}\frac{\partial(\gamma T)}{\partial x_k} \tag{11.35}$$

The corresponding volume–averaged energy equation for the solid medium can be written in a similar way as for the fluid medium (subscript s denotes solid):

$$\frac{\partial}{\partial t}(\gamma_s\rho_s H_s) = \frac{\partial}{\partial x_j}\left(\gamma_s q^e_{s,j}\right) + \frac{1}{V}\int_{A_i}\lambda_s\frac{\partial \widetilde{T}_s}{\partial x_j}n_{s,j}dA + \gamma_s\rho_s q_{s,v},\ \ q^e_{s,j} = \frac{1}{\gamma_s}\lambda^e_{s,jk}\frac{\partial(\gamma_s T_s)}{\partial x_k} \qquad (11.36)$$

where the effective solid conductivity is defined as:

$$\lambda^e_{s,jk} = \lambda_s\delta_{jk} + \lambda^{tor}_{s,jk} \qquad (11.37)$$

Noting that the unit interface surface vector $n_{s,j} = -n_j$, $\gamma_s = 1-\gamma$, and assuming

- constant volume porosity γ,
- the local thermal equilibrium, $T = T_s$ and
- heat flux continuity across the fluid–solid interface area, i.e.

$$\frac{1}{V}\int_{A_i}(\lambda+\lambda_t)\frac{\partial \widetilde{T}}{\partial x_j}n_jdA - \frac{1}{V}\int_{A_i}\lambda_s\frac{\partial \widetilde{T}_s}{\partial x_j}n_jdA = \frac{1}{V}h_iA_i(T_s - T) = 0 \qquad (11.38)$$

(h_i is the interfacial heat transfer coefficient)

the energy conservation for both fluid and solid medium can be derived as follows:

$$\frac{\partial}{\partial t}(\gamma\rho H + (1-\gamma)\rho_s H_s) + \frac{\partial}{\partial x_j}(\gamma\rho H U_j) = \frac{\partial \gamma p}{\partial t} + \frac{\partial}{\partial x_j}\left[\gamma\left(\lambda^{por}_{jk}\frac{\partial T}{\partial x_k} + U_i\left(\tau_{ij}+\tau^t_{ij}\right) + \frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right)\right]$$
$$+\gamma\mathscr{R}_{ij}U^s_jU^s_i + \gamma\rho f_iU_i + \gamma\rho q_v + (1-\gamma)\rho_s q_{s,v} \qquad (11.39)$$

where the effective porous conductivity, λ^{por}_{jk}, is assembled as:

$$\lambda^{por}_{jk} = \left(\lambda + \frac{1-\gamma}{\gamma}\lambda_s + \lambda_t\right)\delta_{jk} + \lambda^{dis}_{jk} + \lambda^{tor}_{por,jk} \qquad (11.40)$$

and the porous tortuosity conductivity, $\lambda^{tor}_{por,jk}$, is introduced to model the tortuosity molecular heat flux:

$$\frac{1}{V}\int_{A_i}(\lambda-\lambda_s)\widetilde{T}n_jdA = \lambda^{tor}_{por,jk}\frac{\partial(\gamma T)}{\partial x_k} \qquad (11.41)$$

□ *Macroscopic turbulent kinetic energy*

If volume averaging is applied to the ensemble/Favre averaged turbulent kinetic energy, Equation (9.19), the intrinsic or volume average of k is given as:

$$k = \frac{1}{2}\langle\widetilde{k}\rangle = \frac{1}{2}\langle\widetilde{u''_iu''_i}\rangle = k_m + \frac{1}{2}\langle\widetilde{(\breve{u}''_i)(\breve{u}''_i)}\rangle,\ \ k_m = \frac{1}{2}\langle\widetilde{u''_i\rangle\langle u''_i}\rangle \qquad (11.42)$$

The modelled transport equation for k can be then derived as (see Nakayama and Kuwahara [1999], Pedras and de Lemos [2001]) by applying the volume average operator to Equation (9.19):

$$\frac{\partial}{\partial t}(\gamma\rho k) + \frac{\partial}{\partial x_j}(\gamma\rho k U_j) = \gamma(P_k + P_b - \rho\,\varepsilon) + \frac{\partial}{\partial x_j}\left[\gamma\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + S_{ex} \qquad (11.43)$$

where the body force production is given by Equation (9.18), while the production due to mean macroscopic flow reads:

$$P_k = -\rho \langle \widetilde{U_i'' U_j''} \rangle \frac{\partial (\gamma U_i)}{\partial x_j} = \tau_{ij}^t \frac{\partial (\gamma U_i)}{\partial x_j} \tag{11.44}$$

and the macroscopic turbulent stress τ_{ij}^t is defined by Equation (11.18). The extra terms appearing in the k-equation, $S_{k,ex}$, are associated with spatial deviations of volume–averaged velocity and turbulent kinetic energy:

$$S_{k,ex} = -\frac{\partial}{\partial x_j} \left(\gamma \overline{\rho} \langle \breve{\tilde{k}} \breve{\tilde{U}}_j \rangle \right) - \gamma \langle \widetilde{U_i'' \breve{U}_j''} \frac{\partial \breve{\tilde{U}}_i}{\partial x_j} \rangle \tag{11.45}$$

These extra terms are usually modelled together. Examples of specific models can be found in Nakayama [2008]; de Lemos and Pedras [2001].

□ *Dissipation of turbulent kinetic energy*

$$\begin{aligned} \frac{\partial}{\partial t} (\gamma \rho \varepsilon) + \frac{\partial}{\partial x_j} (\gamma \rho \varepsilon U_j) &= \gamma \frac{C_{\varepsilon 1} P_k - C_{\varepsilon 2} \rho \varepsilon + C_{\varepsilon 3} P_b}{T_t} \\ -\gamma C_{\varepsilon 4} \rho \varepsilon S_{kk} &+ \frac{\partial}{\partial x_j} \left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \frac{\partial \gamma \varepsilon}{\partial x_j} \right] + S_{\varepsilon,ex} \end{aligned} \tag{11.46}$$

The extra terms which are derived from the presence of porous medium are as follows:

$$S_{\varepsilon,ex} = C_{\varepsilon,ex} \frac{S_{k,ex}}{T_t} - \frac{\partial}{\partial x_j} \left(\gamma \overline{\rho} \langle \breve{\tilde{\varepsilon}} \breve{\tilde{U}}_j \rangle \right) - \gamma C_{\varepsilon 4} \rho \langle \breve{\tilde{\varepsilon}} \frac{\partial \breve{\tilde{U}}_k}{\partial x_k} \rangle + \frac{1}{V} \int_{A_i} \mu \frac{\partial \tilde{\varepsilon}}{\partial x_j} n_j dA \tag{11.47}$$

The current practice is to model the above terms together, see proposals of Nakayama [2008]; de Lemos and Pedras [2001].

11.2 Simplified Modelling Equations

The current implementation of porous modelling is based on the simplified governing equations with the superficial velocity U_i^s being used instead of the intrinsic one, U_i. For all other variables the intrinsic volume averages are employed. Recall that porous media conservation equations are written in terms of intrinsic volume–averaged variables.

Bearing in mind uncertainties in modelling various dispersion terms, tortuosity fluxes and other additional terms arising from volume averaging procedure, these terms have been neglected. Compared to the non-porous equations, the simplified porous equations contain volume porosity γ. Note that the momentum equation is fully implemented, with a general flow resistance model applicable to non–isotropic porous media. The simplified volume–averaged equations read:

□ *Mass conservation*

$$\frac{\partial \gamma \rho}{\partial t} + \frac{\partial}{\partial x_j} (\rho U_j^s) = \gamma s_m \tag{11.48}$$

□ *Momentum conservation*

$$\frac{\partial}{\partial t}\left(\rho U_i^s\right)+\frac{\partial}{\partial x_j}\left(\frac{\rho U_i^s U_j^s}{\gamma}\right)=-\frac{\partial \gamma p}{\partial x_i}+\gamma\rho f_i+\frac{\partial}{\partial x_j}\left[\gamma\left(\tau_{ij}+\tau_{ij}^t\right)\right]+f_{p_i} \tag{11.49}$$

where f_{p_i} represents the resistance force, Equation (11.21)

□ *Species mass fraction conservation*

$$\frac{\partial}{\partial t}\left(\gamma\rho c_i\right)+\frac{\partial}{\partial x_j}\left(\rho c_i U_j^s\right)=\frac{\partial}{\partial x_j}\left[\left(\mathscr{D}_i+\mathscr{D}_{i,t}\right)\frac{\partial\left(\gamma c_i\right)}{\partial x_k}\right]+\gamma s_{c_i} \tag{11.50}$$

□ *Energy conservation*

$$\frac{\partial}{\partial t}\left(\gamma\rho H\right)+\frac{\partial}{\partial x_j}\left(\rho H U_j^s\right)=\frac{\partial \gamma p}{\partial t}+\frac{\partial}{\partial x_j}\left[\gamma(\lambda+\lambda_t)\frac{\partial T}{\partial x_k}+U_i^s\left(\tau_{ij}+\tau_{ij}^t\right)+\gamma\frac{\mu_t}{\sigma_k}\frac{\partial k}{\partial x_j}\right]$$
$$+\gamma\mathscr{R}_{ij}U_j^s U_i^s+\rho f_i U_i^s \tag{11.51}$$

□ *Turbulent kinetic energy and its dissipation rate*

$$\frac{\partial}{\partial t}\left(\gamma\rho k\right)+\frac{\partial}{\partial x_j}\left(\rho k U_j^s\right)=\gamma\left(P_k+P_b-\rho\,\varepsilon\right)+\frac{\partial}{\partial x_j}\left[\gamma\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] \tag{11.52}$$

$$\frac{\partial}{\partial t}\left(\gamma\rho\varepsilon\right)+\frac{\partial}{\partial x_j}\left(\rho\varepsilon U_j^s\right)=\gamma\frac{C_{\varepsilon 1}P_k-C_{\varepsilon 2}\rho\,\varepsilon+C_{\varepsilon 3}P_b}{T_t}$$
$$-\gamma C_{\varepsilon 4}\rho\,\varepsilon S_{kk}+\frac{\partial}{\partial x_j}\left[\gamma\left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] \tag{11.53}$$

As porous structure tends to suppress turbulence, the turbulence modelling is more relevant to predictions of heat and mass transfer than to momentum transfer. In some situations, a simple empirical approach, where k and ε are defined algebraically, is more appropriate instead of solving turbulence transport equations. This approach requires knowledge of porous media turbulence intensity I_{por} and length scale l_{por}:

$$k=\frac{3}{2}\left(U_i^s\right)^2 I_{por}^2,\quad \varepsilon=\frac{C_\mu^{3/4}k^{3/2}}{l_{por}} \tag{11.54}$$

Appropriate values for I_{por} and l_{por} are problem dependent – typical values might be 0.01 and 0.1 × characteristic pore dimension.

11.3 Porous Model Types and User Inputs

The generalised flow resistance Darcy–Forchheimer's model, given by Equation (11.21), is applicable to non–isotropic porous media and includes the viscous, $\mathscr{R}^v_{ij}$, and inertial tensor, $\mathscr{R}^i_{ij}$. These resistance tensors depend on the type of porous structure and have to be specified by the user. Porous model types commonly used in automotive industry and required user's inputs are described next.

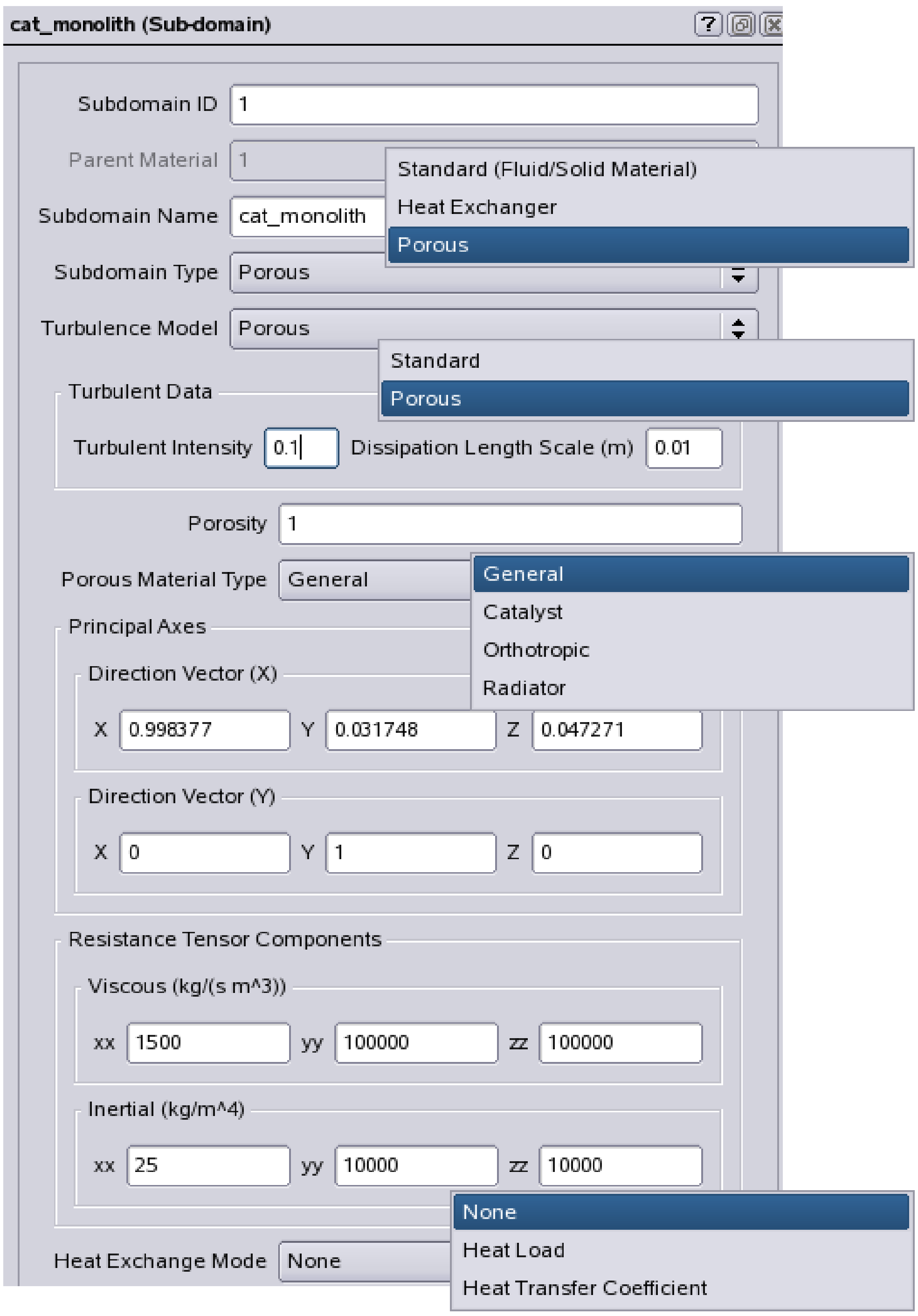
A porous media region has to be associated with the fluid sub–domain; an example is shown earlier in Figure 7.4. The Figure 11.1 shows *R–Desk* Sub–domain panel where Subdomain Name , Subdomain Type and Turbulence Model can be edited or selected. Either Standard or Heat Exchanger or Porous type can be associated with the given fluid sub–domain. The standard sub–domain contains the same continuum as the parent fluid domain and porous or heat exchanger models will not be applied. The heat exchanger type is provided for modelling of the heat transfer between two fluid streams and may or may not have porous structure. The porous sub-domain may or may not involve heat exchange (a radiator always involves heat exchange) which is determined by selecting options from the Heat Exchange Mode list box (shown at bottom of the sub–domain panel).

If the porous turbulence model is selected then the Turbulent Intensity and Dissipation Length Scale values should be supplied inside the Turbulent Data sub–panel. In this case, the turbulence kinetic energy and its dissipation rate will be calculated algebraically from Equation (11.54).

The next step in defining both porous and heat exchanger sub–domain models is to specify the volume Porosity γ and Principal Axes or eigenvectors of the resistance tensor $\mathscr{R}_{ij}$, given by Equation (11.20). Note that the viscous, $\mathscr{R}^v_{ij}$, and inertial, $\mathscr{R}^i_{ij}$, tensors defined along principal axes have only three diagonal coefficients. The basic approach is to specify two direction vectors in a Cartesian coordinate system, i.e. the components X , Y and Z for the Direction Vector (X) and Direction Vector (Y) , see the Principal Axes sub–panel. Whilst these two direction vectors have to be orthogonal, they *need not to be aligned* with the domain Cartesian system or mesh lines. If the user fail to specify two normal direction vectors, the solver will ensure their orthogonality by modifying the second direction vector. In the case of non–aligned principal and coordinate axes, the direction vectors might not be known a priory, and the user can use the *R–Desk* geometry tools to determine the direction vectors. Among various Porous Material Types : General , Catalyst , Orthotropic and Radiator , only the General type as well as Heat Exchanger require specification of the second Direction Vector Y . When the second direction vector is not required it is determined by solver as well as the third vector (principal axes) which is normal to the plane defined by the first two direction vectors.

The 2nd order resistance tensors specified by user are defined in the principal axes coordinate system and need to be transformed from the principal axes to the solver's (global Cartesian) axes. This is done with the help of transformation matrix $\mathbf{T}_p$. This matrix is the transpose of the matrix $\mathbf{P}_{ax}$ which is user–defined by components of three principal axes vectors, i.e. $\mathbf{T}_p = \mathbf{P}^T_{ax}$. Denoting user–defined viscous and inertial principal (diagonal) resistance tensors as $\mathbf{R}^v_p$ and $\mathbf{R}^i_p$, respectively, they are transformed into the solver's Cartesian axes tensors, $\mathbf{R}^v = \mathscr{R}^v_{ij}$ and $\mathbf{R}^i = \mathscr{R}^i_{ij}$, as

$$\mathbf{R}^v = \mathbf{T}_p\,\mathbf{R}^v_p\,\mathbf{T}^T_p\,,\;\mathbf{R}^i = \mathbf{T}_p\,\mathbf{R}^i_p\,\mathbf{T}^T_p \tag{11.55}$$



cat_monolith (Sub-domain)

Subdomain ID: 1
Parent Material: 1
Subdomain Name: cat_monolith
Subdomain Type: Porous
Turbulence Model: Porous

Turbulent Data
Turbulent Intensity: 0.1
Dissipation Length Scale (m): 0.01

Porosity: 1
Porous Material Type: General

Principal Axes
Direction Vector (X): X 0.998377, Y 0.031748, Z 0.047271
Direction Vector (Y): X 0, Y 1, Z 0

Resistance Tensor Components
Viscous (kg/(s m^3)): xx 1500, yy 100000, zz 100000
Inertial (kg/m^4): xx 25, yy 10000, zz 10000

Heat Exchange Mode: None

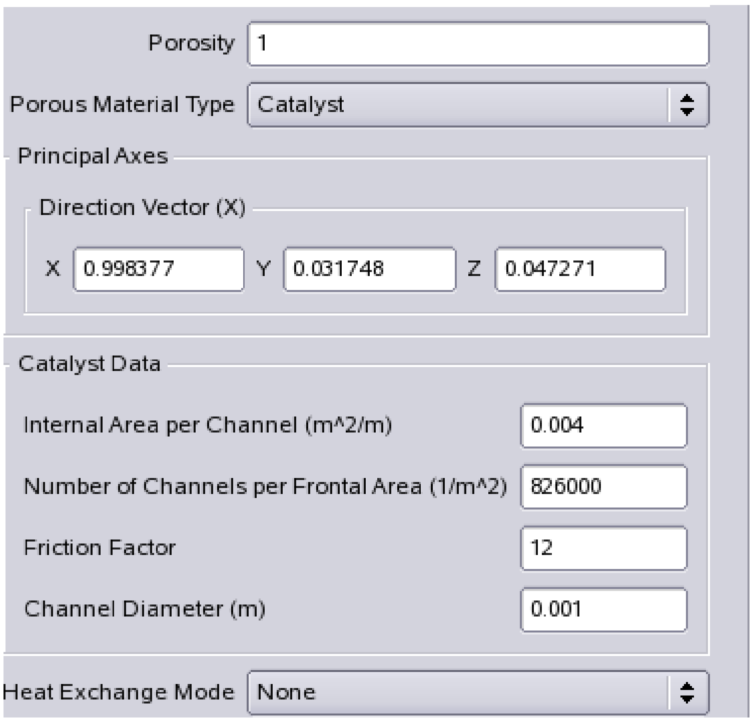
Figure 11.1: *R–Desk* Sub–domain panel. User inputs for general porous media modelling.

11.3.1 General porous model

If a general porous material or heat exchanger has been selected by the user the diagonal elements of Viscous and Inertial resistance tensors $\mathbf{R}_p^v$ and $\mathbf{R}_p^i$, respectively, need to be provided for each principal direction xx , yy and zz , see sub–panel Resistance Tensor Components in Figure 11.1. Large resistance values in certain direction ($\approx$ 50,00 to 100,000 for viscous and $\approx$ 10,000 for inertial resistance) will suppress the flow in that direction. For example values in Figure 11.1, the flow will be directed along the first principal axes, i.e along the Direction Vector X axes.

11.3.2 Catalytic converter

The flow through the catalytic converter is assumed to be in the direction of the first principal axes, i.e. in the Direction Vector X . The flow direction is enforced in the solver by using large resistance values in other two principal directions. Also, the resistance is modelled according to Darcy's viscous model where the pressure drop depends linearly on the local velocity magnitude. Thus the inertial resistance is neglected. The viscous resistance tensor $\mathbf{R}_p^v$ is then derived from the Catalyst Data depicted in Figure 11.2. This data include specification of Internal Area per Channel



Porosity	1
Porous Material Type	Catalyst

Principal Axes

Direction Vector (X)

X	Y	Z
0.998377	0.031748	0.047271

Catalyst Data

Internal Area per Channel (m^2/m)	0.004
Number of Channels per Frontal Area (1/m^2)	826000
Friction Factor	12
Channel Diameter (m)	0.001

Heat Exchange Mode: None

Figure 11.2: *R–Desk* Sub–domain panel. User inputs for the catalyst porous model.

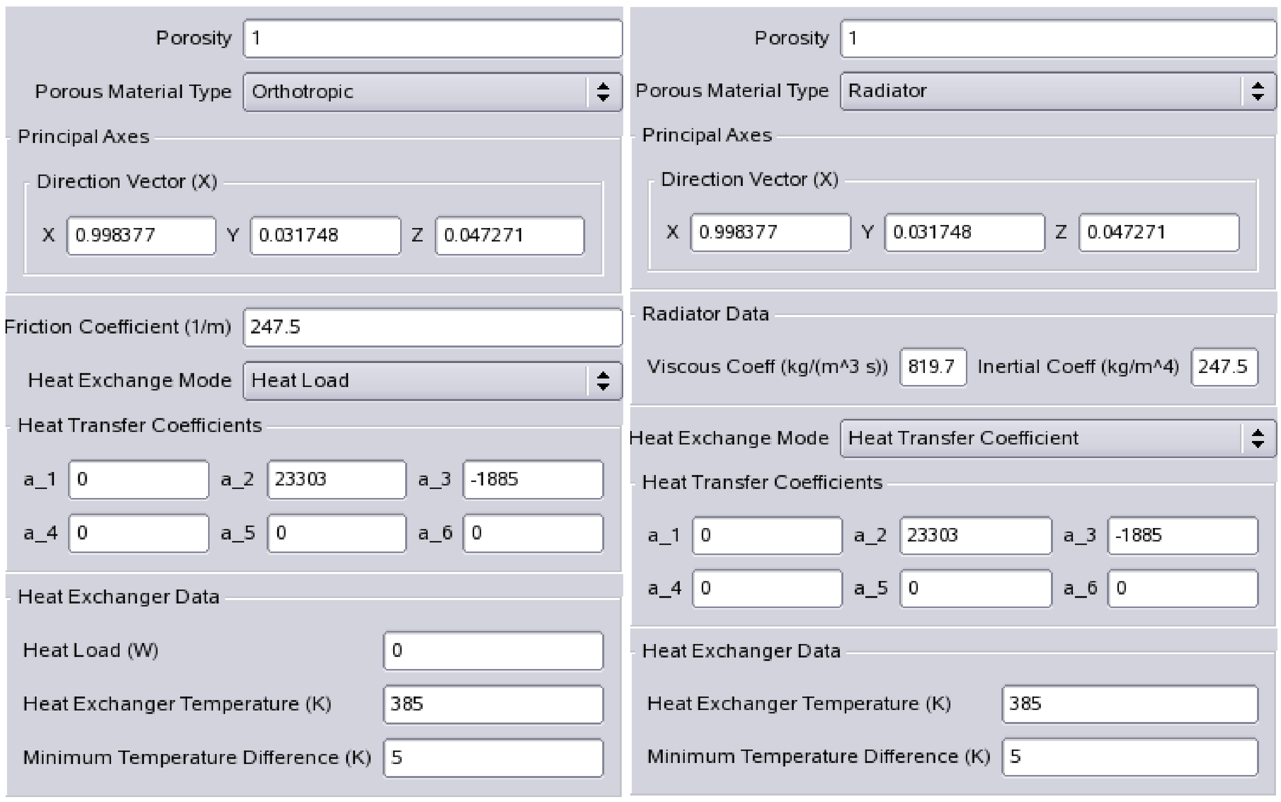
, Number of Channels per Frontal Area , Friction Factor and Channel Diameter . Based on the above data, the viscous resistance in the flow direction is calculated as:

$$R_{p,x}^v = \frac{A_c N_c f_c \mu}{\gamma D_c} \, [kg/(m^3 s)] \tag{11.56}$$

where A_c, N_c, f_c and D_c signify internal area per unit length of a single catalyst pore (channel), number of channels per frontal catalyst area, non-dimensional friction coefficient and hydraulic of a channel, respectively. The molecular viscosity of fluid is denoted as μ.

11.3.3 General orthotropic model

The general orthotropic model also enforces uni–directional flow in direction of the first principal axis, Figure 11.3 (left). The viscous resistance is neglected, and the inertial resistance is calculated



Porosity: 1
Porous Material Type: Orthotropic
Principal Axes
Direction Vector (X): X 0.998377 Y 0.031748 Z 0.047271
Friction Coefficient (1/m): 247.5
Heat Exchange Mode: Heat Load
Heat Transfer Coefficients
a_1 0 a_2 23303 a_3 -1885
a_4 0 a_5 0 a_6 0
Heat Exchanger Data
Heat Load (W): 0
Heat Exchanger Temperature (K): 385
Minimum Temperature Difference (K): 5

Porosity: 1
Porous Material Type: Radiator
Principal Axes
Direction Vector (X): X 0.998377 Y 0.031748 Z 0.047271
Radiator Data
Viscous Coeff (kg/(m^3 s)): 819.7 Inertial Coeff (kg/m^4): 247.5
Heat Exchange Mode: Heat Transfer Coefficient
Heat Transfer Coefficients
a_1 0 a_2 23303 a_3 -1885
a_4 0 a_5 0 a_6 0
Heat Exchanger Data
Heat Exchanger Temperature (K): 385
Minimum Temperature Difference (K): 5

Figure 11.3: *R–Desk* Sub–domain panel. User inputs for the orthotropic (left) and radiator (right) porous model.

from the knowledge of the dimensional Friction Coefficient f_{ort}, $1/m$:

$$R^{i}_{p,x} = \frac{1}{2}\rho f_{ort} \; [kg/m^4] \tag{11.57}$$

where ρ denotes the fluid density.

11.3.4 Radiators

The radiator sub–domain type is intended to model heat exchange in automotive radiators where the sub–domain fluid represents a hot stream, and it is cooled by the cold fluid stream of infinite thermal capacitance. This *single stream heat exchange model* accounts for the fluid sub–domain stream explicitly while the second stream is assumed to have a constant temperature, T_{ref}, specified by the user.

The flow through a radiator is uni–directional and both viscous, $R^{v}_{p,x}$, and inertial, $R^{i}_{p,x}$, resistances are specified via Viscous Coeff and Inertial Coeff edit boxes, respectively, which are shown in Figure 11.3 (right).

The Heat Exchange Mode can be either Heat Transfer Coefficient or Heat Load . If the heat load mode is selected then the value of Heat Load (the total heat exchange Q_{hx}) should be specified inside the Heat Exchanger Data sub–panel, see Figure 11.3. When the heat transfer coefficient mode is enabled, the heat load Q_{hx} will be calculated by the solver. In both cases, the volume

Heat Transfer Coefficient h_{hx} should be define as a polynomial function of the superficial velocity magnitude U^s via six coefficients a_1 to a_6 . Thus, h_{hx} will be calculated as:

$$h_{hx} = a_1 + a_2 U^s + a_3 (U^s)^2 + a_4 (U^s)^3 + a_5 (U^s)^4 + a_6 (U^s)^5 \tag{11.58}$$

After defining the heat transfer coefficient, the Heat Exchanger Temperature T_{ref} and the Minimum Temperature Difference ΔT_{min} have to be provided in the Heat Exchanger Data sub–panel. The minimum temperature difference is defined as the difference between the minimum sub-domain fluid temperature $T_{f,min}$ and the heat exchanger (radiator) reference temperature T_{ref}, i.e. $\Delta T_{min} = T_{f,min} - T_{ref} > 0$ if the sub–domain fluid is cooled. If the sub-domain fluid is heated (heater) $\Delta T_{min} = T_{ref} - T_{f,max} > 0$ where $T_{f,max}$ denotes maximum sub-domain fluid temperature.

As a radiator is a heat exchanger more details about the heat exchange modelling is given in the next section.

11.3.5 Heat exchangers

For a general heat exchanger, the flow resistance can be specified as for the general porous model.

In terms of heat exchange, heat exchangers can be classified as:

- *Heaters:* Sub-domain fluid is heated, reference heat exchanger temperature T_{ref} should be greater than the maximum fluid temperature $T_{f,max}$ for at least ΔT_{min} where T_{ref} and ΔT_{min} are specified by the user inside Heat Exchanger Data group box, Figure 11.3.
- *Coolers:* Sub-domain fluid is cooled, i.e. $T_{ref} < T_{f,min} - |\Delta T_{min}|$.

For both heat exchange modes (either Heat Load or Heat Transfer Coefficient), the Heat Transfer Coefficients h_{hx} are provided as for the radiators For the given $h_{hx} = F(U^s)$, it is possible that maximum/minimum sub-domain fluid temperature goes above/below the reference temperature T_{ref} for the heater/cooler, respectively. The solver takes care about these non-physical model situation by adjusting heat exchanger reference temperature T_{ref} depending on the exchanger type. The exchanger type (cooler or heater) is not currently explicitly specified. It can be, however, specified through values of the minimum temperature difference ΔT_{min} as follows:

- $\Delta T_{min} > 0$: heater, $T_{ref} \geq T_{f,max} + \Delta T_{min}$
- $\Delta T_{min} < 0$: cooler, $T_{ref} \leq T_{f,min} - |\Delta T_{min}|$
- $\Delta T_{min} = 0$: allows both heating and cooling.

12 MODELLING MULTIPHASE FLOWS

This chapter introduces basic physical and modelling aspects of multiphase flow and describes two modelling approaches: multi–fluid or Euler–Euler approach and single–fluid Mixture and VOF approach together with corresponding modelling equations. After that setting of available models is presented.

12.1 Introduction

Multi–phase flow can be defined as the simultaneous flow of several phases where the phase is the one of the three thermodynamic states of matter: either solid or liquid or gas. In many industrial flow environments, multiphase flows are very common – they are generally the rule rather than the exception. Typical examples in the automotive industry are engine injection systems (diesel fuel injectors), spray combustion, coolant systems, fuel thanks (filling, sloshing), crank-cases with oil, cavitating fuel pumps.

In terms of physical states, a two–phase flow can be classified as liquid–gas, solid–gas, liquid–solid and liquid–liquid (immiscible liquids). In terms of topology, Ishii [1975] sub–divides multi–phase flows into:

- □ dispersed (air–ice flow, oil droplets in water or air, gas bubbles in liquid, solid particles in gas/liquid)
- □ separated (annular liquid/gas flows, free surface flows, water jets in air)
- □ and mixed (transitional) (bubbly/droplet annular flows, slug/churn liquid and gas flows).

Typical multiphase topology is illustrated in Figure12.1 for flow boiling in a heated tube with a sub-cooled inlet.

The Instantaneous Navier–Stokes equations describe both single and multi–phase flows. This description is not restricted to either laminar/turbulent regime or dispersed/separated topology. However, the resolving of scales associated with the topology of phase interfaces or with the fluid (turbulent) motion is not yet computationally affordable. Instead, two basic modelling approaches are employed in practice:

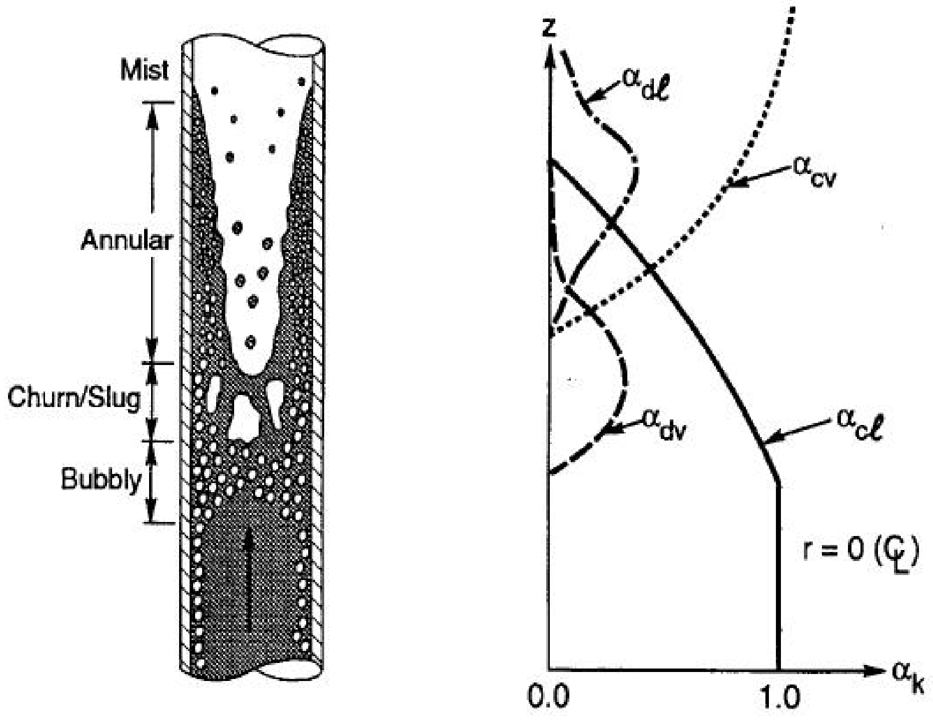


Figure 12.1: Typical multiphase topology (flow boiling in a heated tube with a subcool-ed inlet) and volume fraction distributions for four fields: cl– continuous liquid, cv–continuous vapour, dl–dispersed liquid, dv–dispersed vapour. >From Lahey and Drew [2001].

□ *Discrete Phase Modelling (DPM) or Eulerian–Lagrangian approach* – Discrete phase modelling assumes the dispersed topology of the multi–phase flow.

Individual dispersed elements (droplets, bubbles, particles) are tracked in time through the flow domain by solving momentum equations given in the Lagrangian form. The dispersed elements exchange mass, momentum and energy with a continuous fluid phase. The transport equations of the continuous phase are those used in Eulerian–Eulerian approach and they are usually coupled with dispersed element equations. This modelling approach is usually applied to flows with the low volume fraction of dispersed phase elements (less $\approx 10\%$). Spray modelling (atomisers, spray breakups, droplet collisions, dynamic drag accounting for changes in the droplet shape, wall–films, turbulent interactions, etc) belong to this class of multiphase flow. Discrete phase modelling in VECTIS-MAX will be considered in future.

□ *Multi–Fluid Modelling or Eulerian–Eulerian approach* – The Euler–Euler or multi–fluid approach describes each phase by Eulerian conservation equations for mass, momentum and energy. The phases are treated as inter–penetrating continua, with their own field variables such as velocities, pressure and temperature. The space and time (ensemble) averaging of local instantaneous phasic equations requires a phase indicator function (phase fraction) which is equivalent to the volume (void) fraction of the corresponding phase in case of steady–state flows. In general, the phase fraction, which has been introduced in the section dealing with properties of multiphase mixture, is the probability that a certain phase exists at a certain point in space and time. The averaging process introduces, in addition to the turbulent Reynolds stresses or diffusion fluxes, unknown extra terms into basic equations. These terms describe inter–phase mass, momentum and energy transport and have to be modelled in terms of known base variables.

The Reynolds–stress and inter–phase transport closure problem is the crux of multi-fluid methodology since this closure depends on the nature of the flow, i.e. it relies on empirical and problem dependent input. However, the Euler–Euler modelling can be applied to all multiphase flow regimes and its predictive capabilities are satisfactory in many application areas (cf. Lahey and Drew [2001]).

The remainder of this section deals only with the Euler–Euler multi–phase modelling.

12.2 Euler–Euler Modelling Equations

The averaging of instant multiphase flow equations can be done in different ways. The mass (density) weighted averaging gives a simple form of conservation equations. For example, the ensemble average velocity of the phase k, U_i^k, is based on the weighting the instant, local velocity $U_{I,i}^k$ with the local density ρ_I^k:

$$U_i^k = \frac{\overline{\rho_I^k U_{I,i}^k}}{\overline{\rho_I^k}} = \frac{\overline{\rho_I^k U_{I,i}^k}}{\rho^k}. \tag{12.1}$$

For single–phase turbulent compressible flows, the density–weighted ensemble (Favre) averaging results with the RANS equations. Similarly, ensemble averaging of local instant phasic equations, Ishii [1975]; Drew [1992], leads to the ensemble mean conservation equations for each phase. These equations are coupled through the phase volume fractions α^k.

The Euler–Euler equations are presented here for non–moving grids and in the framework of eddy–viscosity modelling.

□ *Mass Conservation.*

$$\frac{\partial \alpha^k \rho^k}{\partial t} + \frac{\partial}{\partial x_j}\left(\alpha^k \rho^k U_j^k\right) = \Gamma^k \tag{12.2}$$

Here, superscript k denotes the $k-th$ phase, $k = 1, N_{ph}$, and α^k is the phase volume fraction. The inter–phase mass transfer rate Γ^k occurs for example during evaporation or condensation and needs to be modelled. The compatibility condition for the volume fractions need to enforced:

$$\sum_{k=1}^{N_{ph}} \alpha^k = 1 \tag{12.3}$$

□ *Momentum Conservation.*

$$\frac{\partial}{\partial t}\left(\alpha^k \rho^k U_i^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k \rho^k U_i^k U_j^k\right) = -\frac{\partial\left(\alpha^k p^k\right)}{\partial x_i} + \alpha^k \rho^k f_i + \frac{\partial}{\partial x_j}\left[\alpha^k\left(\tau_{ij}^k + \tau_{ij}^{t,k}\right)\right] + M_i^k \tag{12.4}$$

where the viscous and turbulent stress tensors τ_{ij}^k and $\tau_{ij}^{t,k}$ are given, respectively as:

$$\tau_{ij}^k = 2\mu^k\left(S_{ij}^k - \frac{1}{3}S_{nn}^k \delta_{ij}\right), \quad \tau_{ij}^{t,k} = 2\mu_t^k\left(S_{ij}^k - \frac{1}{3}S_{nn}^k \delta_{ij}\right) - \frac{2}{3}\rho^k k^k \delta_{ij} \tag{12.5}$$

where the phase mean strain tensor is:

$$S_{ij}^k = \frac{1}{2}\left(\frac{\partial U_i^k}{\partial x_j} + \frac{\partial U_j^k}{\partial x_i}\right). \tag{12.6}$$

The term M_i^k represents the inter–phase momentum exchange for the $k-th$ phase and requires modelling.

- □ *Energy Conservation*

$$\frac{\partial}{\partial t}\left(\alpha^k\rho^k H^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k\rho^k H^k U_j^k\right) = \frac{\partial \alpha^k p^k}{\partial t} + \alpha^k\rho^k\left(q_v^k + f_i U_i^k\right) \tag{12.7}$$

$$\frac{\partial}{\partial x_j}\left\{\alpha^k\left[\left(\frac{\mu^k}{Pr^k} + \frac{\mu_t^k}{Pr_t^k}\right)C_p^k\frac{\partial T^k}{\partial x_j} + \frac{\mu_t^k}{\sigma_k^k}\frac{\partial k^k}{\partial x_j} + U_i^k\left(\tau_{ij}^k + \tau_{ij,t}^k\right)\right]\right\} + H_i^k \tag{12.8}$$

where H^k is the total enthalpy, $H^k = C_p^k T^k + U_i^k U_i^k/2 + k^k$, and H_i^k describes the energy inter–phase transfer terms.

- □ *Equation of state.* If a phase is an ideal gas, its density is calculated from the equation of state:

$$\rho^k = \frac{p^k}{R_g^k T^k} \tag{12.9}$$

with R_g^k being the phase gas constant.

- □ *Turbulence Equations.* The turbulent viscosity μ_t^k is predicted in conjunction with the modelled equations for the turbulent kinetic energy k^k and its dissipation rate ε^k. These equations are similar to the single–phase ones, Equations (9.19, 9.20). They read as:

$$\frac{\partial}{\partial t}\left(\alpha^k\rho^k k^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k\rho^k k^k U_j^k\right) = \alpha^k\left(P_k^k + P_b^k - \rho^k\varepsilon^k\right) + \frac{\partial}{\partial x_j}\left[\alpha^k\left(\mu^k + \frac{\mu_t^k}{\sigma_k^k}\right)\frac{\partial k^k}{\partial x_j}\right] + K_i^k \tag{12.10}$$

$$\frac{\partial}{\partial t}\left(\alpha^k\rho^k\varepsilon^k\right) + \frac{\partial}{\partial x_j}\left(\alpha^k\rho^k\varepsilon^k U_j^k\right) = \alpha^k\frac{C_{\varepsilon 1}P_k^k - C_{\varepsilon 2}\rho^k\varepsilon^k + C_{\varepsilon 3}P_b^k}{T_t^k}$$

$$-\alpha^k C_{\varepsilon 4}\rho^k\varepsilon^k S_{nn}^k + \frac{\partial}{\partial x_j}\left[\alpha^k\left(\mu^k + \frac{\mu_t^k}{\sigma_\varepsilon^k}\right)\frac{\partial \varepsilon^k}{\partial x_j}\right] - R^k + \Xi_i^k \tag{12.11}$$

$$\mu_t^k = \rho^k C_\mu \frac{(k^k)^2}{\varepsilon^k} \tag{12.12}$$

In the above equations, K_i^k and Ξ_i^k represent inter–phase turbulent transfer rates for k and ε, respectively.

The unknown inter–phase correlations associated with transport of mass, Γ^k, momentum M_i^k, energy H_i^k and with turbulent quantities K_i^k and Ξ_i^k require closure in terms of dependent variables. The complexities of multi-fluid modelling can be illustrated by work of Kunz et al. [1998]. They analysed boiling flow in a vertical coolant duct by using the four field system, similar to that shown in Figure 12.1. Employing the high Reynolds $k-\varepsilon$ model for two continuous phases and single–pressure model ($p^k = p$) the 25 unknown variables were solved for $(4(U_i^k + H^k + \alpha^k + 2(k^k + \varepsilon^k) + p)$. About 30 inter–phase models were required.

12.3 Multi–Phase Mixture and VOF Models

The multi-fluid model (often referred to as the full Eulerian model) is theoretically well advanced model but also very complex. Alternative and the simpler Mixture and Volume of Fluid (VOF) models are based on the *single fluid* approach. The physical assumption behind these models is that all phases share the same pressure, temperature and turbulent quantities. In case of the Mixture model the phases can move at different velocities relative to the mixture velocity by introducing the concept of *slip velocities*.

While the Mixture model deals with inter–penetrating phases, the VOF model, originally developed by Hirt and Nicholls [1981], is applicable to immiscible fluids, ie. to the capturing of liquid–gas interfaces (free surface flows, jet breakup, large gas bubbles in a liquid, dam break liquid motion). It is a *front capturing* method in a sense that the volume fraction of a phase (liquid) α^k takes the following three values:

- ◻ $\alpha^k = 1 \Rightarrow$ the fluid k occupies completely the cell
- ◻ $\alpha^k = 0 \Rightarrow$ no fluid k in the cell
- ◻ $0 < \alpha^k < 1 \Rightarrow$ the interface between fluid k and other fluids is present.

Predictions of the volume fraction should be accurate enough to get the sharp resolution of the interfaces. This requires the use of High Resolution Interface Capturing (HRIC) schemes and finer numerical grids.

The governing equations can be derived by summing the phase Equations (12.2, 12.4, 12.8, 12.10 and 12.11), and having in mind that the concept of slip velocity is (usually) applied to the momentum and volume fraction equations.

12.3.1 Mixture equations

◻ *Mass conservation for the mixture*

$$\frac{\partial \rho^m}{\partial t} + \frac{\partial}{\partial x_j}\left(\rho^m U_j^m\right) = 0 \tag{12.13}$$

where superscript m designates the mixture. The mixture density and velocities are:

$$\rho^m = \sum_{k=1}^{N_{ph}} \alpha^k \rho^k,\ U_i^m = \frac{\sum_{k=1}^{N} \alpha^k \rho^k U_i^k}{\rho^m}. \tag{12.14}$$

◻ *Momentum conservation for the mixture*

$$\frac{\partial}{\partial t}\left(\rho^m U_i^m\right) + \frac{\partial}{\partial x_j}\left(\rho^m U_i^m U_j^m\right) = -\frac{\partial p}{\partial x_i} + \rho^m f_i$$
$$+\frac{\partial}{\partial x_j}\left(\tau_{ij}^m + \tau_{ij}^{t,m}\right) - \frac{\partial}{\partial x_j}\left(\sum_{k=1}^{N_{ph}} \alpha^k \rho^k U_i^{d,k} U_j^{d,k}\right) \tag{12.15}$$

where the viscous and turbulent stress tensors τ_{ij}^m and $\tau_{ij,t}^m$ are given, respectively as:

$$\tau_{ij}^m = 2\mu^m\left(S_{ij}^m - \frac{1}{3}S_{nn}^m\delta_{ij}\right),\ \tau_{ij}^{t,m} = 2\mu_t^m\left(S_{ij}^m - \frac{1}{3}S_{nn}^m\delta_{ij}\right) - \frac{2}{3}\rho^m k^m\delta_{ij} \tag{12.16}$$

and the mixture mean strain tensor is:

$$S_{ij}^m = \frac{1}{2}\left(\frac{\partial U_i^m}{\partial x_j} + \frac{\partial U_j^m}{\partial x_i}\right). \tag{12.17}$$

Both laminar and turbulent mixture viscosities are phase weighted:

$$\mu^m = \sum_{k=1}^{N_{ph}}\alpha^k\mu^k,\ \mu_t^m = \sum_{k=1}^{N_{ph}}\alpha^k\mu_t^k = \rho^m C_\mu\frac{(k^m)^2}{\varepsilon^m} \tag{12.18}$$

The last term in Equation (12.15) represents momentum diffusion due to the relative motion of phases and contains the drift or diffusion velocities defined as: $U_i^{d,k} = U_i^k - U_i^m$. The diffusion velocity can be determined by employing the algebraic relations for the slip velocity. If the phases move with the same (mixture) velocity (terms containing the drift velocity omitted), the mixture model reduces to a *homogeneous multiphase model*.

□ *The $k-\varepsilon$ model for the mixture*

$$\frac{\partial}{\partial t}\left(\rho^m k^k\right) + \frac{\partial}{\partial x_j}\left(\rho^m k^m U_j^m\right) = P_k^m + P_b^m - \rho^m\varepsilon^m + \frac{\partial}{\partial x_j}\left[\left(\mu^m + \frac{\mu_t^m}{\sigma_k^m}\right)\frac{\partial k^m}{\partial x_j}\right] \tag{12.19}$$

$$\begin{aligned}\frac{\partial}{\partial t}\left(\rho^m\varepsilon^m\right) + \frac{\partial}{\partial x_j}\left(\rho^m\varepsilon^m U_j^m\right) &= \frac{C_{\varepsilon 1}P_k^m - C_{\varepsilon 2}\rho^m\varepsilon^m + C_{\varepsilon 3}P_b^m}{T_t^m}\\ -C_{\varepsilon 4}\rho^m\varepsilon^m S_{nn}^m + \frac{\partial}{\partial x_j}\left[\left(\mu^m + \frac{\mu_t^m}{\sigma_\varepsilon^m}\right)\frac{\partial\varepsilon^m}{\partial x_j}\right] - R^m\end{aligned} \tag{12.20}$$

□ *Energy conservation for the mixture*

$$\begin{aligned}\frac{\partial}{\partial t}\left(\rho^m H^m\right) + \frac{\partial}{\partial x_j}\left(\rho^m H^m U_j^m\right) &= \frac{\partial p}{\partial t} + \rho^m\left(q_v^m + f_i U_i^m\right)\\ +\frac{\partial}{\partial x_j}\left[\left(\frac{\mu^m}{Pr^m} + \frac{\mu_t^m}{Pr_t^m}\right)C_p^m\frac{\partial T^m}{\partial x_j} + \frac{\mu_t^m}{\sigma_k^m}\frac{\partial k^m}{\partial x_j} + U_i^m\left(\tau_{ij}^m + \tau_{ijt}^{t,m}\right)\right]\end{aligned} \tag{12.21}$$

In the above equation the total enthalpy of the mixture H^m takes the usual form: $H^m = C_p^m T^m + U_i^m U_i^m/2 + k^m$, while the specific heat and molecular thermal conductivity of the mixture $\lambda^m = \mu^m C_p^m/Pr^m$ are defined as:

$$C_p^m = \frac{\sum_{k=1}^{N_{ph}}\alpha^k\rho^k C_p^k}{\rho^m},\ \lambda^m = \sum_{k=1}^{N_{ph}}\alpha^k\lambda^k. \tag{12.22}$$

□ *Volume fractions for the mixture*

$$\frac{\partial\alpha^k\rho^k}{\partial t} + \frac{\partial}{\partial x_j}\left(\alpha^k\rho^k U_j^m\right) = -\frac{\partial}{\partial x_j}\left(\alpha^k\rho^k U_j^{d,k}\right) + \Gamma^k \tag{12.23}$$

12.3.2 VOF equations

The VOF model is essentially homogeneous multi-fluid model. Thus, the mixture modelling equations in the previous section describe the VOF model if the terms associated with the drift/slip velocity are omitted. Several HRIC schemes to deliver the sharp interfaces have been designed. The recent one and quite popular is the scheme of Ubbink [1997].

As the VOF method deals with the interfaces between liquid and gas, the surface tension force might have a significant contribution to the inter–phase momentum transfer, depending on the values of the Reynolds ($Re = \rho LU/\mu$, Capillary ($Ca = \mu U/\sigma$) and Weber ($We = \rho LU^2/\sigma$) numbers, where σ is the surface tension, and U, L are the characteristic velocity and length. Generally, the surface tension force is negligible for $Ca >> 1$ or $We >> 1$, where the Ca number is important for $Re << 1$ and the We number for $Re >> 1$.

At a curved interface, the surface tension force is resolved into two components along the unit normal vector n_i (directed from liquid to gas) and along the tangential unit vector t_i:

$$f_{i,\sigma} = f^n_{i,\sigma} n_i + f^t_{i,\sigma} t_i = \sigma K n_i + \frac{\partial \sigma}{\partial t} t_i \tag{12.24}$$

where K is the surface curvature. The surface tension force is usually implemented by using the Continuum Surface Force (CSF) model of Brackbill et al. [1992]. The CSF model can also take into account the effects of wall adhesion in situations when the interface is in contact with the wall.

12.3.3 Setting up the homogeneous mixture model

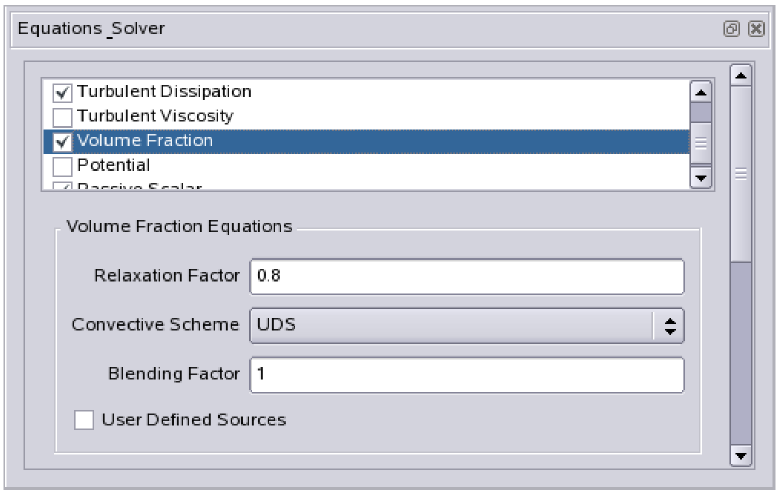
The VECTIS-MAX solver provides the *homogeneous mixture model* which neglects the relative motion between phases, i.e. the drift velocity term in the mixture momentum equation (12.15) is omitted. In addition, other relevant fields such as temperature and turbulence quantities are shared by all phases. The mixture modelling equations are solved together with the volume fraction equations, Equation(12.23, where the interphase mass transfer is neglected ($\Gamma^k = 0$).

Multi–phase modelling approach, i.e. the mixture model, can be enabled within each fluid domain. The actual model selection and definition of phases have been described in Section Setting a multiphase mixture, Figure 8.1. When the multi–phase model is defined the Volume Fraction equation will be activated in the Equations_Solver panel with the default control variable values as Figure 12.2 illustrates.

The computed values of volume fractions have to satisfy the compatibility condition, $\sum_k^{N_{ph}} \alpha^k = 1$. As the phase volume fraction should be bounded, $0 \leq \alpha^k \leq 1$, it could be difficult to satisfy numerically the compatibility condition and ensure the bounded solution. Regarding this, the compatibility condition can be enforced by either solving volume fractions for all phases or solving for $N_{ph} - 1$ phase and calculate the volume fraction of the last phase as

$$\alpha^{N_{ph}} = 1 - \sum_{k=1}^{N_{ph}-1} \alpha^k. \tag{12.25}$$

The Discretise panel, shown in Figure 10.10, provides the Solve All Phases option box to select whether to solve all phase equations or not. The default option is not, i.e. the option box is not active.


Equations_Solver

- [✓] Turbulent Dissipation
- [ ] Turbulent Viscosity
- [✓] Volume Fraction
- [ ] Potential

Volume Fraction Equations

Relaxation Factor: 0.8

Convective Scheme: UDS

Blending Factor: 1

[ ] User Defined Sources

Figure 12.2: *R–Desk* setup. Setting solution of the volume fraction equation

12.4 Phase Change Modelling

In automotive industry two types of phase changes are modelled: boiling and cavitation.

12.4.1 Boiling Models

Boiling occurs when the liquid is heated to its saturation temperature (boiling temperature) which results in rapid vaporisation of the liquid. Boiling models discussed in this section are homogenous boiling models and are designed to calculate boiling flow possibly taking place in engine cooling passages. Two boiling models are presented: VECTIS3 model and RPI model. Both models use the homogenous multiphase mixture equations whereby the liquid and vapour phases are solved individually but other fields like temperature, velocities etc are shared between each phase. Recall the mass source Γ^k from equation (12.2). Then if Γ^k_{liquid} and Γ^k_{vapour} are the mass source of the liquid and vapour phases respectively then the following holds:

$$\Gamma^k_{liquid} = -\Gamma^k_{vapour} \tag{12.26}$$

The saturation temperature (T_{sat}) depends on absolute pressure. The absolute pressure is defined in section Reference and solver working pressure

$$p_{aps} = p + p_{ref} + \Delta p_h \tag{12.27}$$

where p is the working pressure, p_{ref} is the reference pressure and Δp_h is the hydrostatic pressure defined in relation 10.15. The set up of body force, reference altitude and reference density needed for calculation of hydrostatic pressure together with reference pressure is given in section Fluid Domain (Figure 17.6).

12.4.1.1 VECTIS3 boiling model

This model was first proposed in Bo [2004] and is based on the assumption that boiling flow is modelled in the nucleate regime. The detailed analysis of the forces acting on bubbles, such as buoyancy, drag, surface tension, inertia, etc., are not included. The drift velocity between the two phases is also not considered.

□ *Volume of Fraction Equation*

For a multi-fluid homogeneous mixture model, the mass conservation Equation (12.2) is re-written in the following form:

$$\frac{\partial \alpha^k}{\partial t} + \frac{\partial}{\partial x_j}\left(\alpha^k U_j^k\right) = \Gamma^k \tag{12.28}$$

where Γ^k is the evaporation/condensation mass source.

□ *Evaporation and Condensation in the bulk*

The source term in Equation (12.28) is expressed as:

$$\Gamma^k = \Gamma^k_{evap} + \Gamma^k_{cond} \tag{12.29}$$

where

$$\Gamma^k_{evap} = max\left[0, \frac{(\bar{\alpha}-\alpha)}{t_{evap}}\right] \tag{12.30}$$

$$\Gamma^k_{cond} = min\left[0, \frac{(\bar{\alpha}-\alpha)}{t_{cond}}\right] \tag{12.31}$$

With $\bar{\alpha}$ defining the reduced thermal equilibrium void fraction $\bar{\alpha}-\alpha$ is the measure of the departure of the current status with respect to the reduced thermal equilibrium, which is considered as the driving force to return to the reduced thermal equilibrium state. t_{evap} and t_{cond} in Equations (12.30) and (12.31) are the timescales for α to return $\bar{\alpha}$ through the processes of evaporation or condensation. In the saturated region this is defined as:

$$\bar{\alpha} = \frac{x^n_{eq}}{x_{eq} + (1-x_{eq})\frac{\rho_g}{\rho_f}} \tag{12.32}$$

where ρ_g and ρ_f are the densities of vapour and liquid. x_{eq} is the vapour mass fraction under the thermal equilibrium condition, which can be calculated according to the enthalpy of the mixture h, the saturated liquid enthalpy h_{sf} and the latent heat of vaporisation h_{fg}:

$$x_{eq} = \frac{h-h_{sf}}{h_{fg}} \tag{12.33}$$

When $n = 1$, $\bar{\alpha}$ becomes the value in thermal equilibrium state. In current work $n = 1.1$ to take effects of thermal non-equilibrium and non-zero drift velocity between two phases into account. In sub-cooled domain, $\bar{\alpha}$ is simply set to zero. The time scale for evaporation t_{evap} is set to a constant in the current study whilst the time scale of condensation t_{cond} is a function of degree of sub-cooling:

$$t_{cond} = K_{cond}\frac{T_{sat}}{\Delta T_{sub}} \tag{12.34}$$

where K_{cond} is a constant, T_{sat} the saturation temperature and ΔT_{sub} the degree of sub-cooling defined as:

$$\Delta T_{sub} = T_{sat} - T \tag{12.35}$$

Enthalpy of the fluid is calculated as follows:

$$h = C_{Pf} min\left[T_{sat}, T\right] + u_{sq} \tag{12.36}$$

where u_{sq} describes the work done by the fluid motion and is calculated as squared velocity vector.

$$u_{sq} = \frac{1}{2} U_j . U_j \tag{12.37}$$

Enthalpy of saturated liquid:

$$h_{sf} = C_{Psf} T_{sat} + u_{sq} \tag{12.38}$$

Enthalpy of saturated gas:

$$h_{sg} = h_{sf} + h_{fg} \tag{12.39}$$

□ *Boiling at a wall surface*

The total wall heat flux is expressed as:

$$q = q_{spl} + q_{nuc} \tag{12.40}$$

where q_{spl} is the single phase heat flux. The nucleate boiling heat flux q_{nuc} is calculated according to Rohsenow [1952] empirical correlation as:

$$q_{nuc} = C_{sf} \mu_f h_{fg} \sqrt{\frac{g(\rho_f - \rho_g)}{\sigma}} \left(\frac{c_{pf} \Delta T_{sup}}{h_{fg} Pr^{1.7}} \right)^{3.03} \tag{12.41}$$

where C_{sf} is an empirical coefficient varying with the liquid-surface combination, g is gravity, σ is the surface tension, ΔT_{sup} is the degree of super-heating. Other quantities like c_{pf} and Pr represent the specific heat and Prandtl number of the liquid phase respectively. The coefficient ΔT_{sup} is given by:

$$\Delta T_{sup} = T_{wall} - T_{sat} \tag{12.42}$$

where T_{wall}, T_{sat} and T are heated wall temperature, the local saturation temperature and mean flow temperature respectively.

□ *Evaporation rate from the wall*

Using the total heat flux q from relation 12.40 the evaporation rate from the wall is then calculated as:

$$\Gamma^k = m_v \frac{10^{-7} q}{h_{fg} + C_{pf} \Delta T_{sub}} \tag{12.43}$$

with m_v calculated from:

$$m_v = \frac{1}{\rho_l (x\_mass + ratio\_d(1 - x\_mass)^2)} \tag{12.44}$$

and x_mass following for relation:

$$x\_mass = r_d max(10^{-6}, \alpha)/(1 - \alpha + r_d \alpha) \tag{12.45}$$

The ratio density r_d is given by:

$$r_d = \frac{\rho_g}{\rho_l} \tag{12.46}$$

12.4.1.2 RPI boiling model

The RPI model is mainly used in Euler-Euler multi-phase flows but was adopted for use in homogenous multi-phase mixture flows. In the RPI model the vapour bubble diameter and quenching heat flux are taken into account. The mass flow from the wall is calculated directly from the nucleate boiling heat flux.

- *Volume of Fraction Equation*

  The mass conservation equation (12.2) is used in its exact form.

- *Evaporation and Condensation in the bulk*

  The evaporation/condensation rate in the bulk flow is given as:

$$\Gamma^k = \frac{h_i A_i \left(T - T_{sat}\right)}{h_{fg}} \tag{12.47}$$

where A_i is the interfacial area density and is calculated as:

$$A_i = 6\frac{lsg(1-\alpha)}{1-lsg}\frac{1}{d_{bulk}} \tag{12.48}$$

where $lsg = min\left(\alpha, 0.25\right)$ according to Kurul and Podowski [1990] and d_{bulk} is the bulk bubble diameter and is calculated as a function of local sub-cooling as follows (Kurul and Podowski [1990]):

$$d_{bulk} = \begin{cases} 1.510^{-4} & ; \quad \Delta T_{sub} > 13.5K \\ 1.510^{-3} - 10^{-4}\Delta T_{sub} & ; \quad 0 < \Delta T_{sub} < 13.5K \\ 1.510^{-3} & ; \quad \Delta T_{sub} < 0 \end{cases}$$

The heat transfer coefficient h_i in relation 12.47 is expressed as:

$$h_i = \frac{Nu\lambda_l}{d_{bulk}} \tag{12.49}$$

The Nusselt number Nu is calculated according to Ranz and Marshall [1952] correlation as:

$$Nu = 2 + 0.6Re^{\frac{1}{2}}Pr^{\frac{1}{3}} \tag{12.50}$$

The Reynolds and Prandtl numbers are evaluated as:

$$Re = \frac{\rho_l U d_{bulk}}{\mu_l} \tag{12.51}$$

$$Pr = \frac{\mu_l C_{pl}}{\lambda_l} \tag{12.52}$$

The quantity U in Equation (12.51) is the mixture velocity magnitude.

□ *Boiling at a wall surface*

The formulation of the mechanistic model takes into account the single phase heat flux q_{spl}, the nucleate boiling heat flux q_{nuc} and the quenching heat flux q_{que} to give the total heat flux:

$$q = q_{spl} + q_{nuc} + q_{que} \tag{12.53}$$

The nucleate boiling heat flux q_{nuc} is given by the following relation:

$$q_{nuc} = (VfN_a)\,\rho_g h_{fg} \tag{12.54}$$

where V is the bubble volume at departure from the wall and is evaluated as:

$$V = \frac{D_{bw}^3 \pi}{6} \tag{12.55}$$

A number of models have been proposed to calculate the wall bubble diameter D_{bw} in relation 12.55. A modified version of the Tolubinsky and Konstanchuk [1970] model is employed as:

$$D_{bw} = min\left(1.2exp\left(-\frac{\Delta T_{sub}}{45\,[K]}\right)[mm]\,, 1.4\,[mm]\right) \tag{12.56}$$

The bubble release frequency in relation 12.54 is given by the following expression:

$$f = \sqrt{\frac{4g(\rho_l - \rho_g)}{3\rho_l Dbw}} \tag{12.57}$$

whereas the nucleation site density Na is given according to **?** as:

$$N_a = 210\,(T_{wall} - T_{sat})^{1.805} \tag{12.58}$$

The quenching heat flux in relation 12.53 is evaluated by using the following expression:

$$q_{que} = \alpha_q \Omega (T_{wall} - T) \tag{12.59}$$

where Ω is the effective wall area covered with bubbles and is calculated as:

$$\Omega = min\left(one, 0.25Dbw^2 \pi N_a \eta\right) \tag{12.60}$$

The dimensionless number η is given as $\eta = 4.8e^{\frac{-Ja}{80}}$. The Jacob number J_a is defined according to (Kenning and Victor [1981]) as follows:

$$J_a = \frac{C_{pl}\rho_l \Delta T_{sat}}{\rho_g h_{fg}} \tag{12.61}$$

The quenching heat transfer coefficient α_q in relation 12.59 is calculated as:

$$\alpha_q = 2\sqrt{\frac{C_q f \lambda_l \rho_l C_{pl}}{\pi}} \tag{12.62}$$

where λ_l is the conductivity of the liquid and C_q is a constant with a value of 0.8.

□ *Evaporation rate from the wall*

Mass flow rate from the wall is now calculated using relation 12.54 for the nucleate boiling heat flux q_{nuc}:

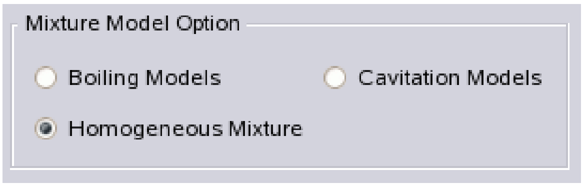
$$\Gamma^k = \frac{q_{nuc}}{h_{fg} + C_{pf}\Delta T_{sub}} \tag{12.63}$$

12.4.1.3 Setting up a Boiling Model

Two phases need to be defined so that boiling model can be used: the liquid and vapour phases. It is recommended that the compressibility for the vapour phase should be set to weakly compressible.

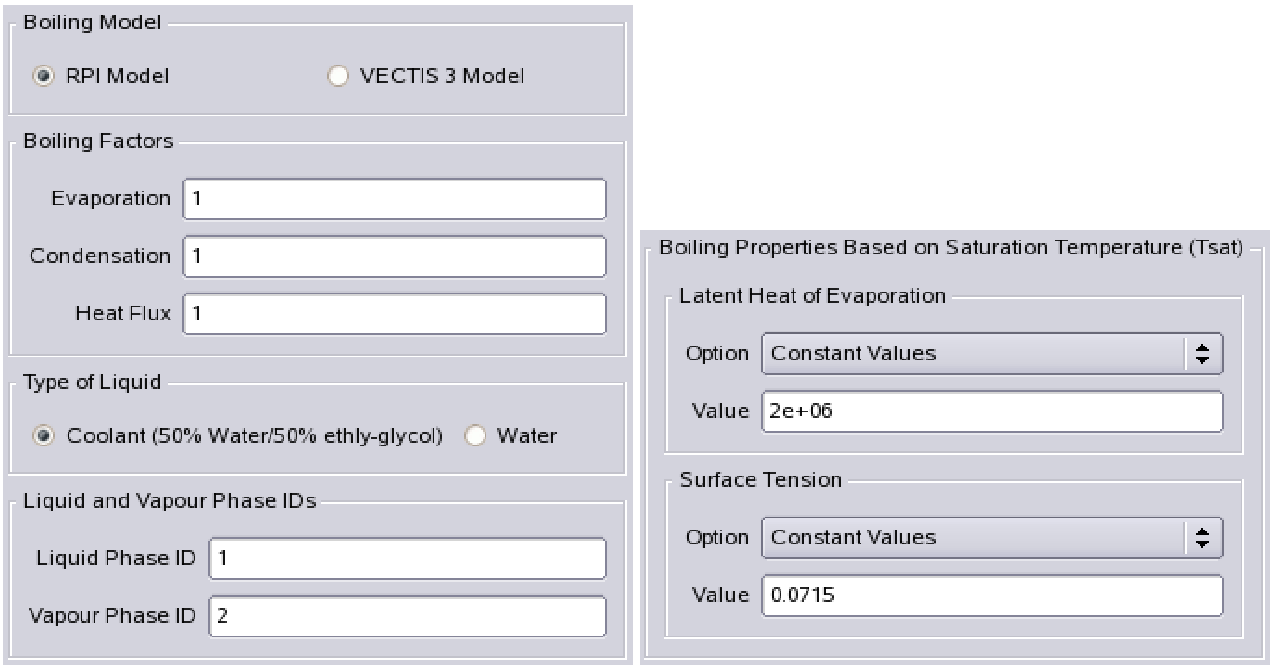
To set up a boiling model, select the appropriate Fluid Domain from the Solver Setup Tree . This opens up the Fluid Domain set up panel. Under Multiphase Modelling select the Mixture option (see section 8.6.1). As a result a Multiphase option is added to the Solver Setup Tree . Left-clicking this option opens the Multiphase panel. Under Mixture Model Option select Boiling Models as in Figure 12.3. A number of fields become active. Either RPI model or VECTIS3 model can be selected. The effect of evaporation, condensation and boiling heat flux can either be increased or decreased by entering the appropriate values for Evaporation , Condensation and Heat Flux . The liquid phase can either be Water or Coolant (50% water; 50% ethyl-glycol) . This selection determines the method used to calculate the saturation temperature (Tsat). Since the Fluid Domain can have any number of phases, the phase ID for the 2 phases (liquid and vapour) used in boiling have to be selected by entering the ID number of the phase in Liquid Phase ID and Vapour Phase ID respectively.

Calculation Option for properties such as Latent Heat of Evaporation and Surface Tension can either be Constant Values or Polynomial f(Tsat) (available from the drop down menu). In case of


Mixture Model Option

Boiling Models Cavitation Models [✓ Homogeneous Mixture]

Figure 12.3: *R–Desk* setup for multiphase mixture models


Boiling Model

[✓ RPI Model] VECTIS 3 Model

Boiling Factors

Evaporation	1
Condensation	1
Heat Flux	1

Type of Liquid

[✓ Coolant (50% Water/50% ethly-glycol)] Water

Liquid and Vapour Phase IDs

Liquid Phase ID	1
Vapour Phase ID	2

Boiling Properties Based on Saturation Temperature (Tsat)

Latent Heat of Evaporation

Option	Constant Values
Value	2e+06

Surface Tension

Option	Constant Values
Value	0.0715

Figure 12.4: *R–Desk* setup for boiling models and parameters

polynomial option these two properties are dependent on saturation temperature.

12.4.2 Cavitation Models

The modelling of cavitation if very significant in engineering designs. Cavitation can have a profound effect on many engineering systems and its careful simulation is crucial. Some of the examples where cavitation plays an important role include: fuel injectors, hydrostatic bearings, marine propellers and many others.

When cavitation occurs on a device it affect performance (such as load symmetry, noise, vibration, reduced flow rate etc.) and may cause physical damage to the device.

Thus physical understanding of cavitation and implementation are very important to industrial applications so that cavitation can be eliminated during the design stages.

The cavitation model presented here is a multiphase homogenous mixing of the liquid and vapour phases and the slip velocity between phases is neglected. The onset of cavitation occurrence is associated with the saturation pressure in the liquid at a given temperature which causes cavitation bubbles to generate. These cavitation bubbles then grow and eventually collapse in a short time to release huge amount of energy associated with very high temperature and pressure on very small surfaces.

Cavitation was first studied by Lord Rayleigh in the 19th century (Rayleigh [1917]). More sophisticated models that can be applied to a vast number of cavitating problems have been developed recently.

□ *Volume of Fraction Equation*

Recall equation 12.2 for a multi-fluid homogeneous mixture model where the mass conservation for each phase is given as:

$$\frac{\partial \alpha^k \rho^k}{\partial t} + \frac{\partial}{\partial x_j}\left(\alpha^k \rho^k U_j^k\right) = \Gamma^k \tag{12.64}$$

where superscript k denotes the $k-th$ phase, and α^k is the phase volume fraction.

The mass source Γ^k in relation 12.28 is given by:

$$\Gamma^k = \Gamma^k_{evap} + \Gamma^k_{cond} \tag{12.65}$$

where Γ^k_{evap} and Γ^k_{cond} represent the evaporation and condensation rates. These phase change rates for cavitation are derived from Rayleigh-Plesset and are presented here for a number of models.

12.4.2.1 Singhal et al. Model:

The cavitation model presented here builds on the full cavitation model proposed in Singhal et al. [2002].

The vapour mass fraction and fluid density is given by:

$$\frac{1}{\rho} = \frac{f_v}{\rho_v} + \frac{1-f_v}{\rho_l} \tag{12.66}$$

where ρ_l, ρ_v are the density of liquid and vapour respectively. Similarly, f_v represent the mass fraction of vapour.

The vapour volume fraction α_v is derived as:

$$\alpha_v \equiv f_v \frac{\rho}{\rho_v} \tag{12.67}$$

The phase change rates Γ^k_{evap} and Γ^k_{cond} are approximated as follows:

$$\Gamma^k_{evap} = C_e \frac{\sqrt{k}}{\sigma} \rho_l \rho_v \left[\frac{2}{3}\frac{P_v - P}{\rho_l}\right]^{1/2} \left(1 - \frac{\rho_v \alpha_v}{\rho}\right) \text{if } P_v > P \tag{12.68}$$

$$\Gamma^k_{cond} = C_c \frac{\sqrt{k}}{\sigma} \rho_l \rho_v \left[\frac{2}{3}\frac{P - P_v}{\rho_l}\right]^{1/2} \left(\frac{\rho_v \alpha_v}{\rho}\right) \text{if } P_v < P \tag{12.69}$$

where C_{evap} and C_{cond} are empirical constants and their recommended values are 0.02 and 0.01 respectively, σ is the surface tension of the liquid. The phase change threshold pressure P_v in relation (12.68) and (12.69) is estimated by taking into account the effect of turbulence on cavitating flow:

$$P_v = P_{sat} + 0.5P'_{turb} \tag{12.70}$$

where P_{sat} is the liquid saturation vapour pressure and P'_{turb} represents the local values of the turbulent pressure fluctuations and is estimated as:

$$P'_{turb} = 0.39\rho k \tag{12.71}$$

where k is the local turbulent kinetic energy. To account for laminar flow set the minimum for $\sqrt{k}$ to 1. The turbulence contribution in relation 12.70 is neglected ($P_v = P_{sat}$).

12.4.2.2 Zwart-Gerber-Melamri model:

The Zwart et al. [2004] model is based on the assumption of a constant bubble size. The phase change sources Γ^k_{evap} and Γ^k_{cond} are calculated as:

$$\Gamma^k_{evap} = C_{evap} \frac{3\alpha_{nuc}(1-\alpha_v)\rho_v}{R_B} \left[\frac{2}{3}\frac{P_v - P}{\rho_l}\right]^{1/2} \text{if } P_v > P \tag{12.72}$$

$$\Gamma^k_{cond} = C_{cond} \frac{3\alpha_v \rho_v}{R_B} \left[\frac{2}{3}\frac{P_v - P}{\rho_l}\right]^{1/2} \text{if } P_v < P \tag{12.73}$$

where C_{evap} and C_{cond} are the evaporation and condensation coefficients with values 50 and 0.01 respectively, α_{nuc} is the nucleation site volume density and is given the value $5x10^{-4}$, R_B is the bubble radius and is given the value of $10^{-6}m$.

12.4.2.3 Schnerr-Sauer model:

The Schnerr and Suaer [2001] model uses a variable bubble radius diameter $\mathscr{R}_B$ directly in phase change sources and is defined as:

$$\mathscr{R}_B = \left(\frac{\alpha_v}{1-\alpha_v} \frac{3}{4\pi} \frac{1}{\eta} \right)^{1/3} \tag{12.74}$$

The optimal value for bubble number density η is in the region of 10^{13}.

The phase change sources Γ^k_{evap} and Γ^k_{cond} are then defined as:

$$\Gamma^k_{evap} = C_{evap} \frac{\rho_v \rho_l}{\rho} \alpha_v (1-\alpha_v) \frac{3}{\mathscr{R}_B} \left[\frac{2}{3} \frac{P_v - P}{\rho_l} \right]^{1/2} \quad \text{if } P_v > P \tag{12.75}$$

$$\Gamma^k_{cond} = C_{cond} \frac{\rho_v \rho_l}{\rho} \alpha_v (1-\alpha_v) \frac{3}{\mathscr{R}_B} \left[\frac{2}{3} \frac{P - P_v}{\rho_l} \right]^{1/2} \quad \text{if } P_v < P \tag{12.76}$$

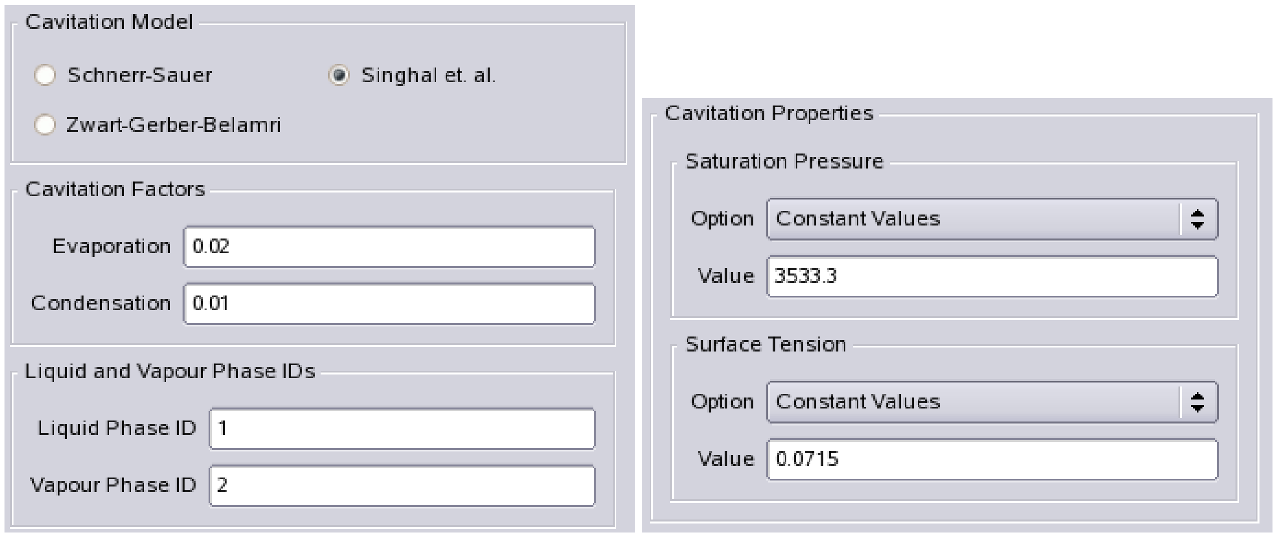
Both C_{evap} and C_{cond} are given values of 1.0 for this model.

12.4.2.4 Setting up a Cavitation Model

Two phases need to be defined as in the boiling case: the liquid and vapour phases. It is recommended that the compressibility for the vapour phase should be set to weakly compressible.

To set up a cavitation model, select the appropriate Fluid Domain from the Solver Setup Tree . This opens up the Fluid Domain set up panel. Under Multiphase Modelling select the Mixture option. As a result a Multiphase option is added to the Solver Setup Tree . Left-clicking this option opens the Multiphase panel.

Under Mixture Model Option select Cavitation Models . Three different cavitation models are available: Singhal et al , Zwart-Gerber-Melamri and Schnerr-Sauer .



Cavitation Model

( ) Schnerr-Sauer [✓ Singhal et. al.]
( ) Zwart-Gerber-Belamri

Cavitation Factors

Evaporation	0.02
Condensation	0.01

Liquid and Vapour Phase IDs

Liquid Phase ID	1
Vapour Phase ID	2

Cavitation Properties

Saturation Pressure

Option	Constant Values
Value	3533.3

Surface Tension

Option	Constant Values
Value	0.0715

Figure 12.5: *R–Desk* setup for cavitation models and parameters

The default values for evaporation and condensation reported in literature are loaded automatically for each of the cavitation model. The user may either increase or decrease the evaporation and condensation by entering the appropriate values for Evaporation and Condensation .

Setting up Saturation Pressure is mandatory for all three models. Surface Tension is only required in the Singhal et al model.

13

BOUNDARY & INTERFACE CONDITION TYPES

Two types of boundaries characterise fluids: **walls** and **flow boundaries**. Walls are "natural" boundaries, usually impermeable to fluid flow, and can be stationary or moving. Flow boundaries are introduced to model fluid domains (i.e. to reduce the size of fluid domains) by cutting through the flow. A number of flow boundaries and in turn boundary condition types can be defined depending on the type of flow. In VECTIS-MAX , a boundary region can be associated with the one of the following boundary condition types:

- □ **Velocity Inlet**
- □ **Mass Flow**
- □ **Pressure Inlet or Outlet**
- □ **Stagnation Inlet**
- □ **Flow Outlet**
- □ **Symmetry Plane**
- □ **Wall**

Not all combinations of the above flow boundary condition types are physically compatible. With regards to the flow outlets, within the given fluid domain:

1. the flow outlet boundaries with the specified flow split and fixed mass flow rate can not coexist.
2. the flow outlet with specified mass flow rate must be used with at least one pressure inlet/outlet.
3. the flow outlet with the prescribed flow split can not coexist with pressure boundaries.

In case of compressible fluid flows, the recommended combinations are given as a function of the flow type (Mach number) throughout a fluid domain:

□ **Subsonic flow:** $Ma < 1$ within the fluid domain

Inflow	*Outflow*
Stagnation Inlet	Pressure Outlet (specified pressure)
Mass Flow Inlet	Pressure Outlet (specified pressure)
Pressure Inlet, $Ma < 0.3$	Pressure Outlet (specified pressure)

□ **Transonic flow:** $Ma < 1$ and $Ma > 1$ within the fluid domain

Subsonic Inflow	*Subsonic Outflow*
Stagnation Inlet	Pressure Outlet (specified pressure)
Mass Flow Inlet	Pressure Outlet (specified pressure)
Subsonic Inflow	*Supersonic Outflow*
Stagnation Inlet	Pressure Outlet (all variables extrapolated)
Supersonic Inflow	*Supersonic Outflow*
Stagnation Inlet (all variables specified)	Pressure Outlet (all variables extrapolated)

□ **Supersonic flow:** $Ma > 1$ within the fluid domain

Supersonic Inflow	*Supersonic Outflow*
Stagnation Inlet (all variables specified)	Pressure Outlet (all variables extrapolated)

Boundary regions associated with solid materials can have either **Wall** or **Symmetry Plane** condition type.

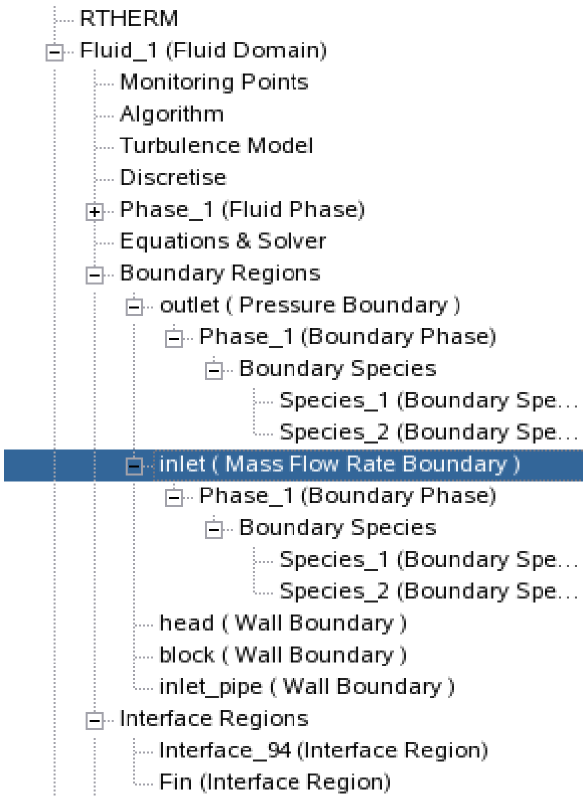
A boundary condition type along Interface Regions is relevant for fluid/solid interfaces where wall boundary type is applicable to the solution of fluid flow equations. The exception is the energy equation for which all the interfaces are treated in an implicit manner, i.e. no specific boundary types are required.

13.1 Setting Up Boundary Condition Types

When starting a new VECTIS-MAX project in *R–Desk* a Boundary region is created within a Fluid Domain by default as Bnd_Reg_1 (Inlet, Given Velocity Boundary) . The initial Solver Setup Tree is given in Figure 7.2 (top left Figure). This boundary region contains Phase_1 (Boundary Phase) definitions that is also added by default.

R–Desk Setup allows the user to add or delete boundary regions. For example, the user can add a new boundary region by right-clicking on the current boundary region and then selecting Add Boundary Region which is the first option in the panel triggered by right-clicking. When there are 2 or more boundary regions, *R–Desk* creates a new node called Boundary Regions which contains all boundary regions defined for that fluid domain. Multiple boundary regions can be added by left-clicking on Boundary Regions . A panel is displayed to the right that contains a button Add Child Boundary Regions . Left-clicking this button adds more boundary regions and gives default names. A boundary region can also be removed by right-clicking on the relevant boundary region and selecting for example the Delete Bnd_Reg_1 (Inlet, Given Velocity Boundary) option. The automatic extraction of boundary and interface regions described in Section 7.5 is currently a simple and efficient method to import already defined boundary and interface regions from the numerical grid.

A boundary and interface region sub–trees (a part of Solver Setup Tree), listing typical boundary types for the Fluid Domain , are illustrated in Figure 13.1. The Solid Domain is described by either Wall Boundary or Symmetry Boundary . In case of the Multi-Component Phase, *R–Desk* adds by default the node Boundary Species for the following boundary condition types: Inlet, Given Velocity , Mass Flow Rate , Pressure and Stagnation types. The node Boundary Species contains all species defined for each Fluid Phase in the Solver Setup Tree (see also Figure 8.3).



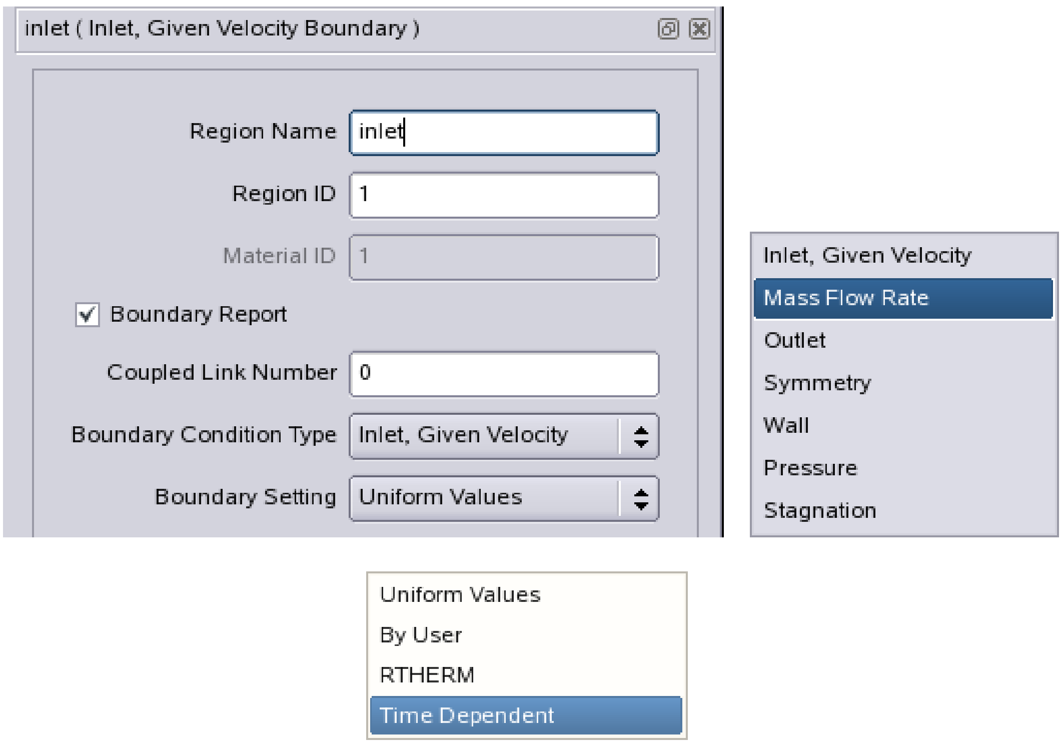
```
RTHERM
Fluid_1 (Fluid Domain)
    Monitoring Points
    Algorithm
    Turbulence Model
    Discretise
    Phase_1 (Fluid Phase)
    Equations & Solver
    Boundary Regions
        outlet ( Pressure Boundary )
            Phase_1 (Boundary Phase)
                Boundary Species
                    Species_1 (Boundary Spe...
                    Species_2 (Boundary Spe...
        inlet ( Mass Flow Rate Boundary )
            Phase_1 (Boundary Phase)
                Boundary Species
                    Species_1 (Boundary Spe...
                    Species_2 (Boundary Spe...
        head ( Wall Boundary )
        block ( Wall Boundary )
        inlet_pipe ( Wall Boundary )
    Interface Regions
        Interface_94 (Interface Region)
        Fin (Interface Region)
```

Figure 13.1: Boundary and interface region nodes in the *R–Desk* Solver Setup Tree .

Similarly, if passive scalars are defined for Phase1 (Fluid Phase) then Boundary Passive Scalar node will be added.

A boundary condition type and the region attributes can be changed by left-clicking on the Bnd_-Reg... and as a result the Inlet, Given Velocity Boundary panel is displayed to the right as in Figure 13.2 (left). For each boundary region the following attributes are displayed:

- Region Name - can be any given meaningful name
- Region ID - each region has a unique ID by which it can be identified
- Material ID - each region belongs to a particular material.
- Boundary Report - this option is used to output results related to boundary regions. For example if a boundary is a wall then using this option will output wall force coefficients to ASCII files with extension '.wall'.
- Coupled Link Number for 1D/3D coupled simulation with WAVE code (used for Fluid Domains only).
- Boundary Condition Type .
- Boundary Setting .


inlet (Inlet, Given Velocity Boundary)

Region Name: inlet
Region ID: 1
Material ID: 1
[✓] Boundary Report
Coupled Link Number: 0
Boundary Condition Type: Inlet, Given Velocity
Boundary Setting: Uniform Values

Figure 13.2: Boundary condition setup panel and boundary condition types in *R–Desk*

Left-click on Boundary Condition Type will open up the ListBox that contains all boundary condition types as shown in Figure 13.2 (right). Similarly, left-click on Boundary Setting displays currently available setting options which are depicted at the bottom of the above figure. These options are:

- ☐ provision of region–wise Uniform Values
- ☐ specification of boundary values By User which implies the use of User boundary conditions routine.
- ☐ definition of wall thermal conditions by using Ricardo's *R–Therm* module, option RTHERM . The *R–Therm* module generates thermal conditions at various wall boundaries which represent typical engine components used in conjugate heat transfer simulations.
- ☐ provision of Time Dependent boundary values

Depending on the selected boundary condition type the definition of additional attributes might be required. The following sub–sections describe inputs relevant to a specific boundary condition type.

13.2 Velocity Inlet

The fluid velocity vector and other scalars (with exception of pressure) are prescribed at the inlet. The velocity inlet should not be used for sub–sonic compressible flows.

Empirical relations are used to estimate the turbulent kinetic energy in absence of experimental data. The knowledge of the relative free-stream turbulence intensity can be used to estimate the turbulent kinetic energy at inlet, k_b, as:

$$I_\infty = \frac{u'}{u_\infty} = \frac{\sqrt{\bar{u}^2}}{U_\infty} \Rightarrow k_b = \frac{3}{2}(I_\infty U_b)^2 \tag{13.1}$$

Subscript ∞ denotes the free-stream conditions. Using Kolmogorov relation, the inlet values for the dissipation rate, ε_b, can be estimated by:

$$\varepsilon_b \approx \frac{C_\mu^{3/4} k_b^{3/2}}{L_\varepsilon} \tag{13.2}$$

where $L_e/C_\mu^{3/4}$ represents a fraction of the characteristic (inlet) dimension (one-tenth of the shear layer width or the domain size). Furthermore, ε_b may be defined by using the ratio of the turbulent and molecular viscosity at the inlet as follows:

$$\varepsilon_b = \frac{\rho_b C_\mu k_b^2}{\mu} \left(\frac{\mu_t}{\mu}\right)^{-1} \tag{13.3}$$

The values of 1 to 10 for μ_t/μ can be often used at inlet boundaries representing the free-stream boundaries.

13.2.1 Setting up a velocity inlet boundary condition

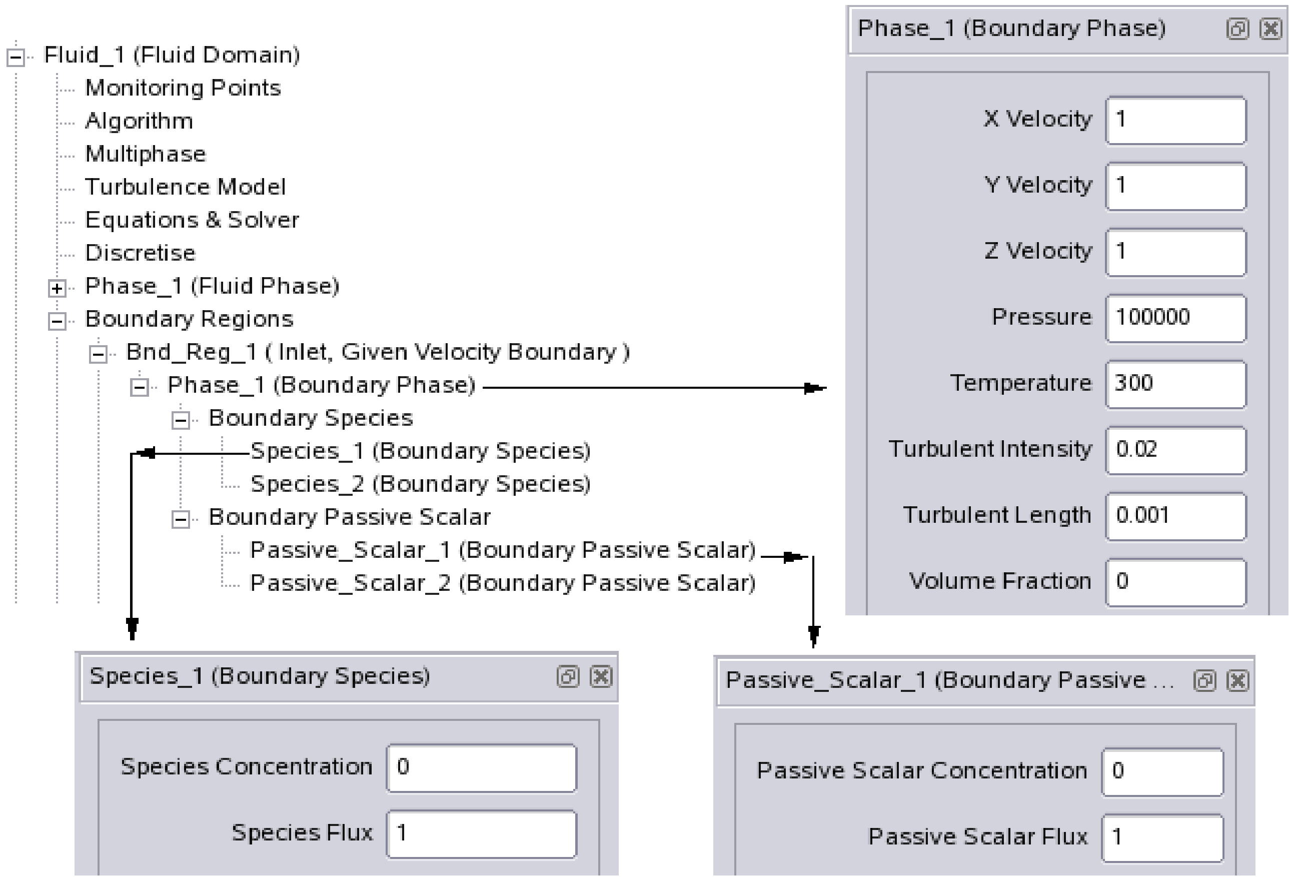
The setup panel for the Velocity Inlet boundary condition is given in Figure 13.2. In order to enter variable boundary values for each phase, the Boundary Phase setup panel shown in Figure 13.3 (top right) should be open by selecting the particular Boundary Phase from the Solver Setup Tree, Figure 13.3 (top left). In a similar way, Boundary Species and Boundary Passive Scalar panels are displayed as illustrated in Figure 13.3 (bottom left) and (bottom right), respectively.

Phase Values: The inlet velocity values are entered in terms of Cartesian velocity components labelled with X Velocity, Y Velocity and Z Velocity. Next, Temperature and estimated *absolute* Pressure are specified. A boundary value of Volume Fraction is required when modelling multiphase flow, i.e. when the volume fraction equation is solved for. If a turbulent flow is simulated then boundary conditions for turbulent flow include Turbulent Intensity and Turbulent Length.

Species Values: In case of a multi–component phase (made of two or more species) boundary values of Species Concentration (mass fractions) for each species should be specified. The Species Flux value is not used at inlets.

Passive Scalar Values: The Passive Scalar Concentration value is required when solving the passive scalar transport equation. The Passive Scalar Flux value is not used at inlets.

Note that the boundary values of volume fractions and species/passive scalar concentrations (mass fractions) must be between zero and one and that their sum over all phases or species/passive scalars should be one.



```
Fluid_1 (Fluid Domain)
    Monitoring Points
    Algorithm
    Multiphase
    Turbulence Model
    Equations & Solver
    Discretise
    Phase_1 (Fluid Phase)
    Boundary Regions
        Bnd_Reg_1 ( Inlet, Given Velocity Boundary )
            Phase_1 (Boundary Phase)
                Boundary Species
                    Species_1 (Boundary Species)
                    Species_2 (Boundary Species)
                Boundary Passive Scalar
                    Passive_Scalar_1 (Boundary Passive Scalar)
                    Passive_Scalar_2 (Boundary Passive Scalar)
```

Phase_1 (Boundary Phase)

X Velocity	1
Y Velocity	1
Z Velocity	1
Pressure	100000
Temperature	300
Turbulent Intensity	0.02
Turbulent Length	0.001
Volume Fraction	0

Species_1 (Boundary Species)

Species Concentration	0
Species Flux	1

Passive_Scalar_1 (Boundary Passive ...

Passive Scalar Concentration	0
Passive Scalar Flux	1

Figure 13.3: Setting up an Inlet Boundary type; outline of phases, species and passive scalars in *R–Desk*

13.3 Mass Flow

The mass flow rate is usually specified at the inlet to the fluid domain. In case of compressible flows either the magnitude of the normal velocity is scaled or the velocity vector is scaled. For the incompressible flows, the mass flow inlet type is equivalent to the velocity inlet type. For incompressible and weakly compressible flows the mass flow rate can be prescribed at an outflow boundary. In this case at least one pressure inlet or outlet boundary should exist.

The mass flow rate at a boundary face is given as:

$$\dot{m}_b = \rho_b(\vec{u}_b \cdot \vec{A}_b) \tag{13.4}$$

where $\dot{m}_b$ = const.

There are two approaches to maintain the mass flow rate at the boundary constant: *normal velocity scaled* and *velocity vector scaled*. In the first, *normal velocity scaled* approach, the boundary velocity is evaluated as:

$$\vec{u}_b^* = \vec{u}_b + \left(\frac{\dot{m}_b}{\rho_b^*} - \vec{u}_b \cdot \vec{A}_b\right) \cdot \frac{\vec{A}_b}{A_b^2} \tag{13.5}$$

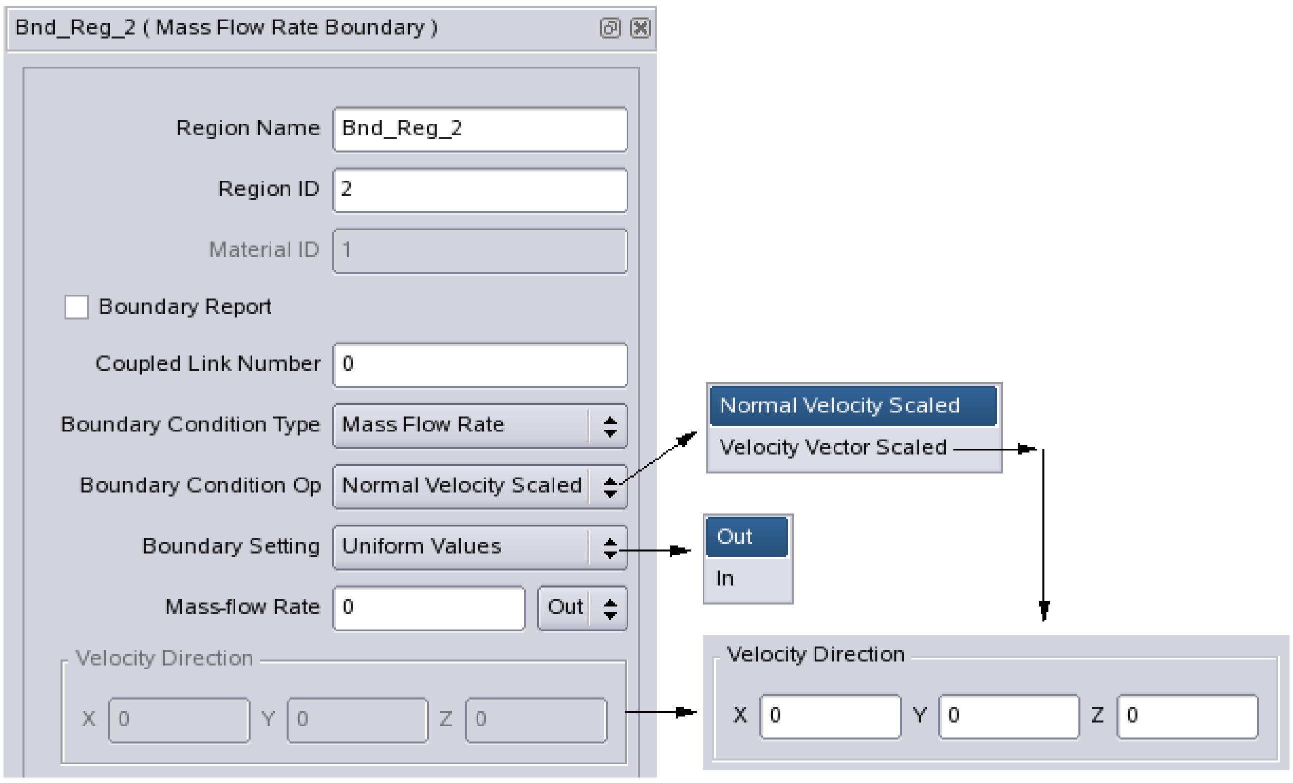
where $\vec{u}_b^*$ is the new iteration value and $\vec{u}_b$ is the value from previous iteration.

In case of the *velocity vector scaled* approach the velocity direction must be specified. The "new" boundary velocity is then evaluated as:

$$\vec{u}_b^* = \frac{\dot{m}_b}{\rho_b^*} \frac{\vec{u}_b}{\left(\vec{u}_b \cdot \vec{A}_b\right)} \tag{13.6}$$

13.3.1 Setting up a mass flow rate boundary condition

A mass flow boundary condition can be specified by left-clicking on the boundary region from the Solver Setup Tree and selecting the Boundary Condition Type from the ListBox to be Mass Flow Rate Boundary . The Mass Flow Rate Boundary setup panel is then displayed as in Figure 13.4. The mass flow rate can be given the appropriate value in Mass-flow Rate InputBox and can have either Out or In attribute. If the In attribute is selected it means the flow enters a fluid domain (inflow boundary), otherwise it leaves a fluid domain (outflow boundary).


Bnd_Reg_2 (Mass Flow Rate Boundary)

Region Name: Bnd_Reg_2
Region ID: 2
Material ID: 1
Boundary Report: ☐
Coupled Link Number: 0
Boundary Condition Type: Mass Flow Rate
Boundary Condition Op: Normal Velocity Scaled
Boundary Setting: Uniform Values
Mass-flow Rate: 0 Out
Velocity Direction: X 0 Y 0 Z 0

Figure 13.4: Setting up a Mass Flow Rate Boundary type in *R–Desk*

There are two Boundary Condition Options available for mass flow rate: Normal Velocity Scaled which is the default selection and Velocity Vector Scaled . When Velocity Vector Scaled is selected then the Velocity Direction is enabled as in Figure 13.4 (right) where Cartesian velocity components should be entered.

Phase, species and passive scalar boundary values are also relevant in the mass flow rate boundary type. Species and passive scalar boundary set up is done in a similar way as for the Inlet, Given Velocity Boundary condition type. To set up the phase boundary values left-click on the appropriate phase under Boundary Phases within Mass Flow Rate Boundary region in the Solver Setup Tree . This opens up the Boundary Phase setup panel similar to Boundary Phase panel in Figure 13.3 (top right figure). The Pressure (an *estimated absolute* value), Temperature and Volume Fraction can be entered into the InputBoxes. In case of a turbulent flow Turbulent Intensity and Turbulent Length should be specified. Compared to the Velocity Inlet type, the X Velocity , Y Velocity and Z Velocity components are not displayed as they are not relevant for the mass flow rate boundary type.

13.4 Pressure Inlet/Outlet

This boundary condition type is based on the known or assumed pressure distribution at boundaries. The constant *static* pressure or boundary *averaged* pressure can be specified at both inflow and outflow boundaries. For incompressible or weakly compressible flows (Mach number $Ma < 0.3$), the *total* pressure p_{tot} can be used at the inlets to internal flows instead of the static pressure p_b. The Stagnation Inlet type is a better choice for the sub–sonic flows ($Ma > 0.3$). The following expression relates the static pressure to the total pressure:

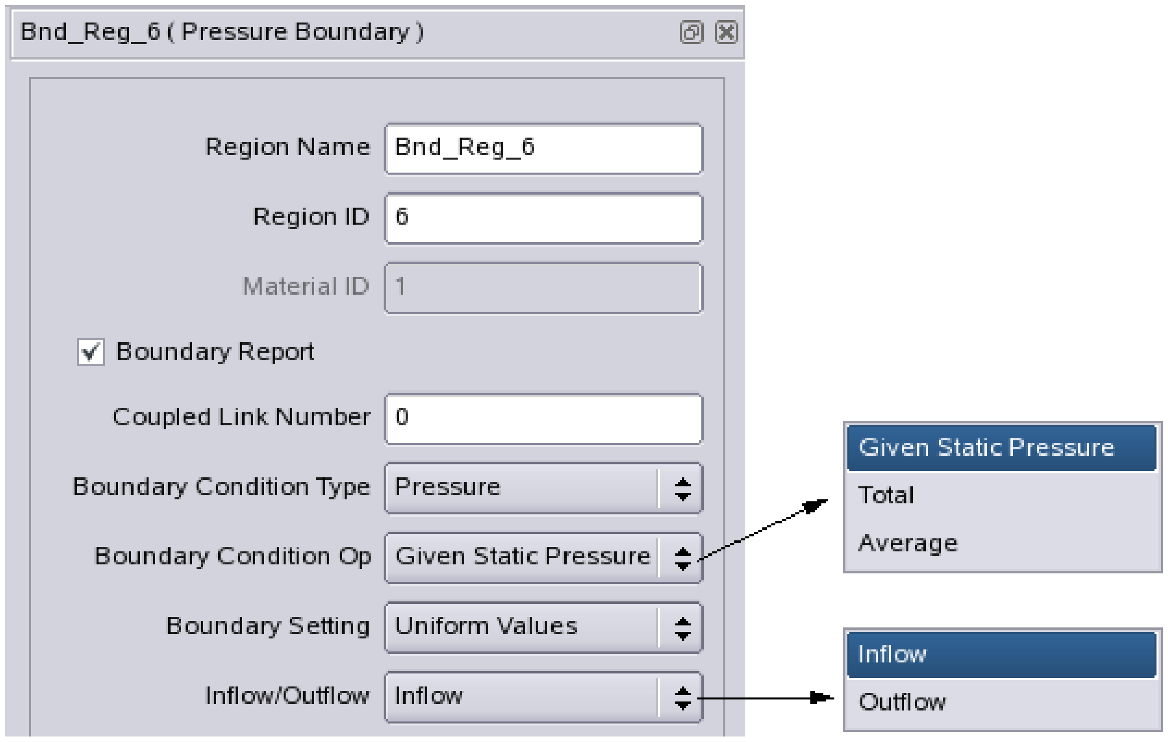
$$p_b = \begin{cases} p_{tot} - \frac{\rho_b |\vec{U}|_b^2}{2} & \text{for inflow} \\ p_{tot} & \text{for outflow} \end{cases} \tag{13.7}$$

The implementation of pressure boundary conditions is discussed in section describing Boundary conditions for pressure corrections. The treatment of other dependent variables such as temperature and turbulence quantities depends on the flow direction. Hence at inflow boundaries the specified inlet values will be used whereas at outflow boundaries all variables will be extrapolated.

13.4.1 Setting up a pressure boundary condition

A pressure boundary can be set up by left-clicking on a boundary region in the Solver Setup Tree and the boundary setup panel is displayed. Under Boundary Condition Type select Pressure and the Pressure Boundary panel with relevant settings is displayed as in Figure 13.5

A number of options is available for the pressure boundary type: Given Static Pressure , Total and Average . For each of these options the boundary condition can be set as Inflow or Outflow under Inflow/Outflow ListBox as indicated in Figure 13.5 (right). Setting up boundary values for phases, species and passive scalars is done in the same way as for to the Mass Flow Rate Boundary type.


Bnd_Reg_6 (Pressure Boundary)

Region Name	Bnd_Reg_6
Region ID	6
Material ID	1
✓ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Pressure
Boundary Condition Op	Given Static Pressure
Boundary Setting	Uniform Values
Inflow/Outflow	Inflow

Figure 13.5: Setting up a Pressure Boundary type in *R–Desk*

13.5 Stagnation Inlet

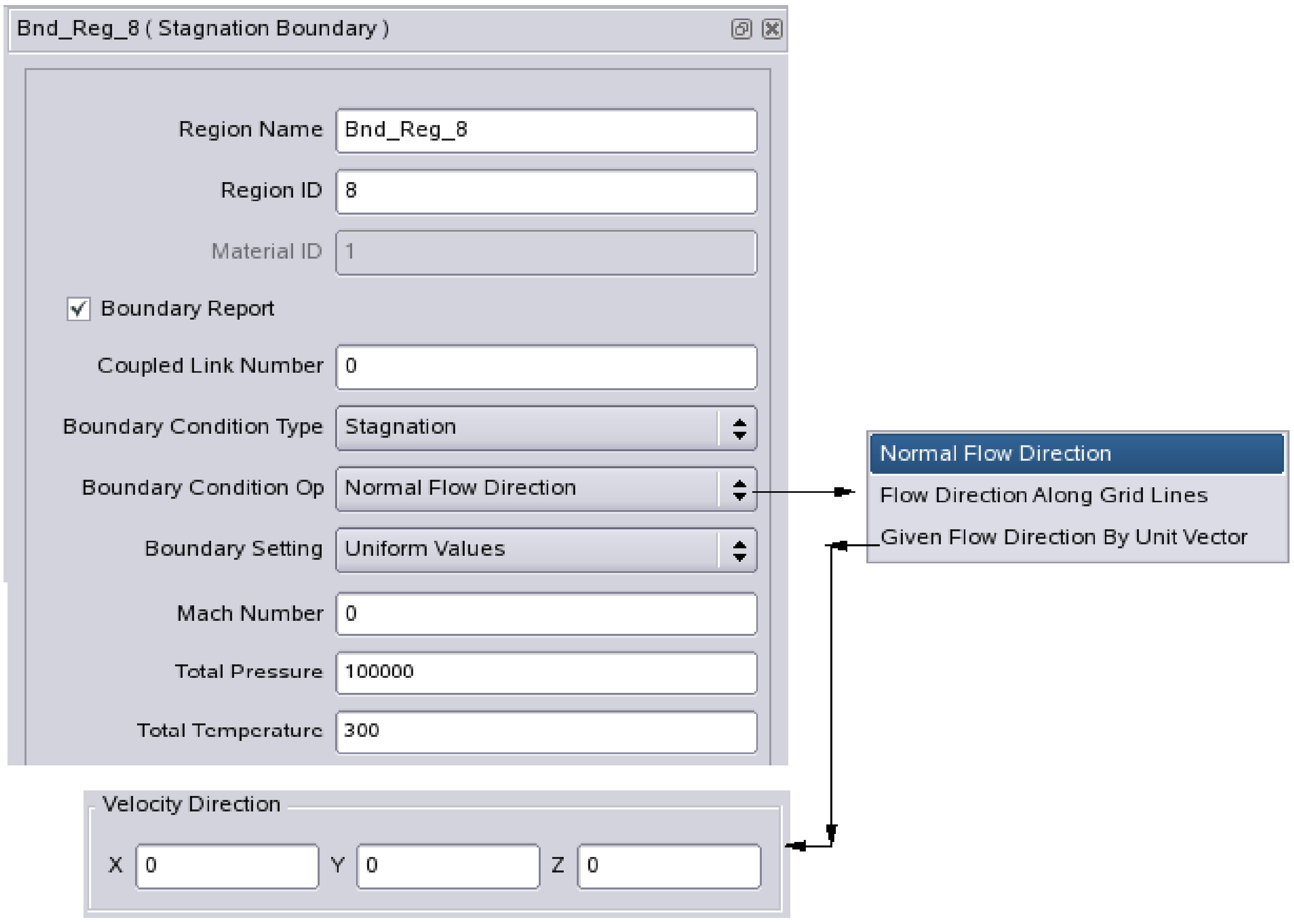
The stagnation inlet describes stagnation or total boundary conditions which correspond to the stagnant fluid (with zero velocity) upstream of the boundary. It should be used for sub–sonic compressible flows. The implementation of this type of boundary is explained in section Subsonic Inlet. Apart from the stagnation pressure (Equation 14.66) and temperature (Equation 14.67) the velocity direction has to be specified.

13.5.1 Setting up a stagnation inlet boundary condition

To set up a Stagnation Boundary, left-click on a boundary region in the Solver Setup Tree to display the boundary condition setup panel. Under Boundary Condition Type select Stagnation option from the ListBox and the Stagnation Boundary panel with relevant settings is displayed as Figure 13.6 shows.

The velocity direction can be specified by selecting one of the following Boundary Condition Op : Normal Flow Direction , Flow Direction Along Grid Lines and Given Flow Direction By Unit Vector . For the latter option Velocity Direction must be defined in terms of the unit vector X,Y,Z components. In the next step, the user should enter values for Mach Number (an estimated value), Total Pressure (an absolute pressure value) and Total Temperature .

Setting up boundary values for phases, species and passive scalars can be done in a similar way as


Bnd_Reg_8 (Stagnation Boundary)

Region Name	Bnd_Reg_8
Region ID	8
Material ID	1
✓ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Stagnation
Boundary Condition Op	Normal Flow Direction
Boundary Setting	Uniform Values
Mach Number	0
Total Pressure	100000
Total Temperature	300

Normal Flow Direction
Flow Direction Along Grid Lines
Given Flow Direction By Unit Vector

Velocity Direction

X	0	Y	0	Z	0

Figure 13.6: Setting up a Stagnation Boundary type in *R–Desk*

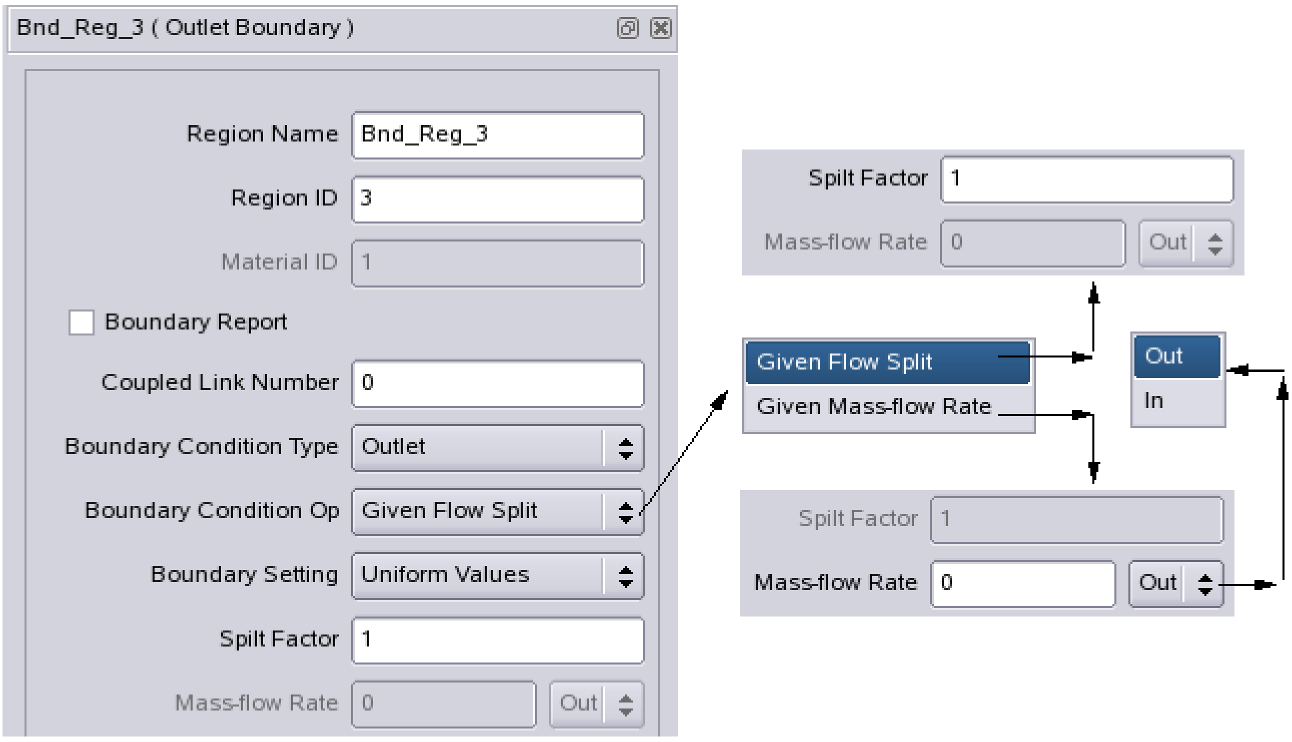
for to the Velocity inlet boundary condition, see also Figure 13.3 (top right). Considering phase variables, only Turbulent Intensity , Turbulent Length and Volume Fraction values might be required.

13.6 Flow Outlet

This type describes outflow boundaries for which fully developed flow conditions can be assumed. In order to comply with the fully developed flow conditions (a zero gradient condition and negligible boundary diffusion flux) the outlet boundaries ought to be placed downstream, far away from the regions with significant flow changes. Two sub–types are available: either prescribed mass flow rate split for single/multiple outlets or prescribed mass flow rate for the single outlet. The flow outlet should not be used for highly compressible flows.

13.6.1 Setting up a flow outlet boundary condition

To set up an Outlet Boundary condition left-click on the relevant boundary region in the Solver Setup Tree . The boundary setup panel is displayed. Under Boundary Condition Type select Outlet from the ListBox and the Outlet Boundary setup panel is displayed. A number of settings becomes available as shown in Figure 13.7.


Bnd_Reg_3 (Outlet Boundary)

Region Name	Bnd_Reg_3
Region ID	3
Material ID	1
☐ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Outlet
Boundary Condition Op	Given Flow Split
Boundary Setting	Uniform Values
Spilt Factor	1
Mass-flow Rate	0 Out

Figure 13.7: Setting up an Outlet Boundary type in *R–Desk*

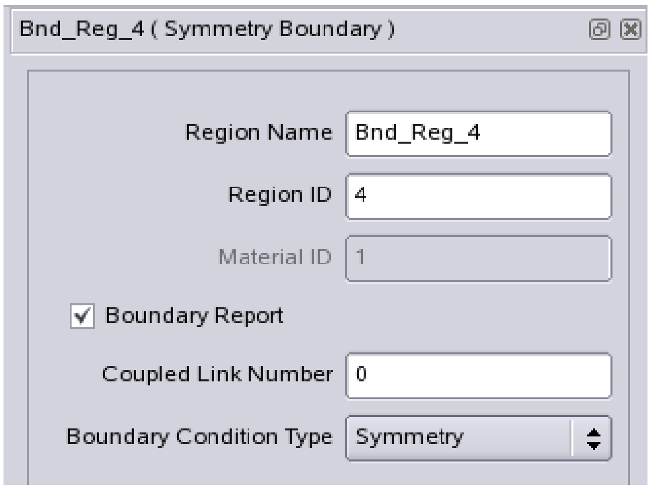
There are two options available for the outlet boundary: Given Flow Split and Given Mass-flow Rate . Figure 13.7 shows the Given Flow Split option where the Split Factor should be entered in the InputBox. When Given Mass-flow Rate option is selected then Split Factor is disabled and Mass-flow Rate is enabled. In this case, the user should always select the Out option as indicated in Figure 13.7 (right).

13.7 Symmetry Plane

The flow symmetry is enforced at this type of boundary by setting to zero the velocity component normal to the symmetry plane. This results in a zero convective flux. Similarly, a zero diffusion flux is obtained by setting to zero the normal derivatives of all other variables.

13.7.1 Setting up a symmetry boundary condition

To set up the symmetry plane condition left-click on the boundary region in the Solver Setup Tree and the boundary set up panel is displayed to the right. Under Boundary Condition Type select Symmetry from the ListBox and this sets up a Symmetry Boundary as in Figure 13.8.


Bnd_Reg_4 (Symmetry Boundary)

Region Name: Bnd_Reg_4
Region ID: 4
Material ID: 1
✓ Boundary Report
Coupled Link Number: 0
Boundary Condition Type: Symmetry

Figure 13.8: Setting up a Symmetry Boundary type in *R–Desk*

13.8 Wall

Either *smooth* or *rough* impermeable walls are assumed. For real fluids, the fluid in contact with the wall has the same velocity as the wall. This *no–slip* condition is enforced by specifying the wall velocity components. In case of the inviscid fluid (slip wall) the wall shear stress is zero. Consequently, the velocity normal to the wall must be zero and the tangential component is equal to its counterpart at the near-wall cell centre. These are actually the symmetry plane conditions which means that the *slip wall* can be modelled using the symmetry plane condition type. Note, however, that the symmetry conditions will be enforced for all equations solved for. In case of heat transfer, the wall represents a solid material at which various thermal conditions can be specified such as the prescribed temperature or heat flux.

As the convective fluxes are zero at the wall, only the diffusion fluxes need to be considered in order to predict the wall shear stress, temperature or heat flux distribution. For this, a general expression for the discretised boundary flux, Equation (14.73), is employed, see Implementation of boundary conditions. The wall diffusion coefficients used in the above equation are defined by Equation (9.40). They are valid for both laminar and turbulent flow simulations. For the latter case these are effective wall turbulent coefficients based on the wall function approach, see Section Near-Wall Modelling for more details.

The wall shear stress can be used to calculate the *friction* or *viscous shear* force which acts on the wall boundary region. Thus the total, time-dependent force exerted on the given wall boundary consists of the pressure force $\vec{F}_p$ and friction force $\vec{F}_v$:

$$\vec{F} = \oint_{wall} -p\vec{n}\,dA + \oint_{wall} \tau_{ij}\vec{n}\,dA = \vec{F}_p + \vec{F}_v \,. \tag{13.8}$$

Here p is pressure, τ_{ij} is the wall stress tensor, dA and $\vec{n}$ denote differentially small wall surface and its unit vector, respectively. The total force is usually decomposed into the force parallel to the oncoming stream – the drag force $\vec{F}_D$, and into the force normal to the stream – the lift force $\vec{F}_L$. The forces or force components F_i are usually normalised with the help of reference density ρ_{ref}, velocity U_{ref} and area A_{ref} so that the corresponding force coefficients are defined as:

$$C_{Fi} = \frac{2F_i}{\rho_{ref}U_{ref}^2 A_{ref}} \tag{13.9}$$

The provision of the above reference quantities is explained in Fluid Domain Inputs, Figure 17.6. Both total and viscous force coefficients, defined for each Cartesian coordinate axis, are written by default into the SDF report file. When the Boundary Report box, Figure 13.9, is checked on the force coefficients will be written to the ASCII "wall" file 'projectname.wall_runnumber', see File Output section. Apart from the force coefficients and other variables, the above report files contain the total energy transported through the wall and the area–averaged wall temperature, near–wall temperature and heat transfer coefficient.

13.8.1 Basic wall setup

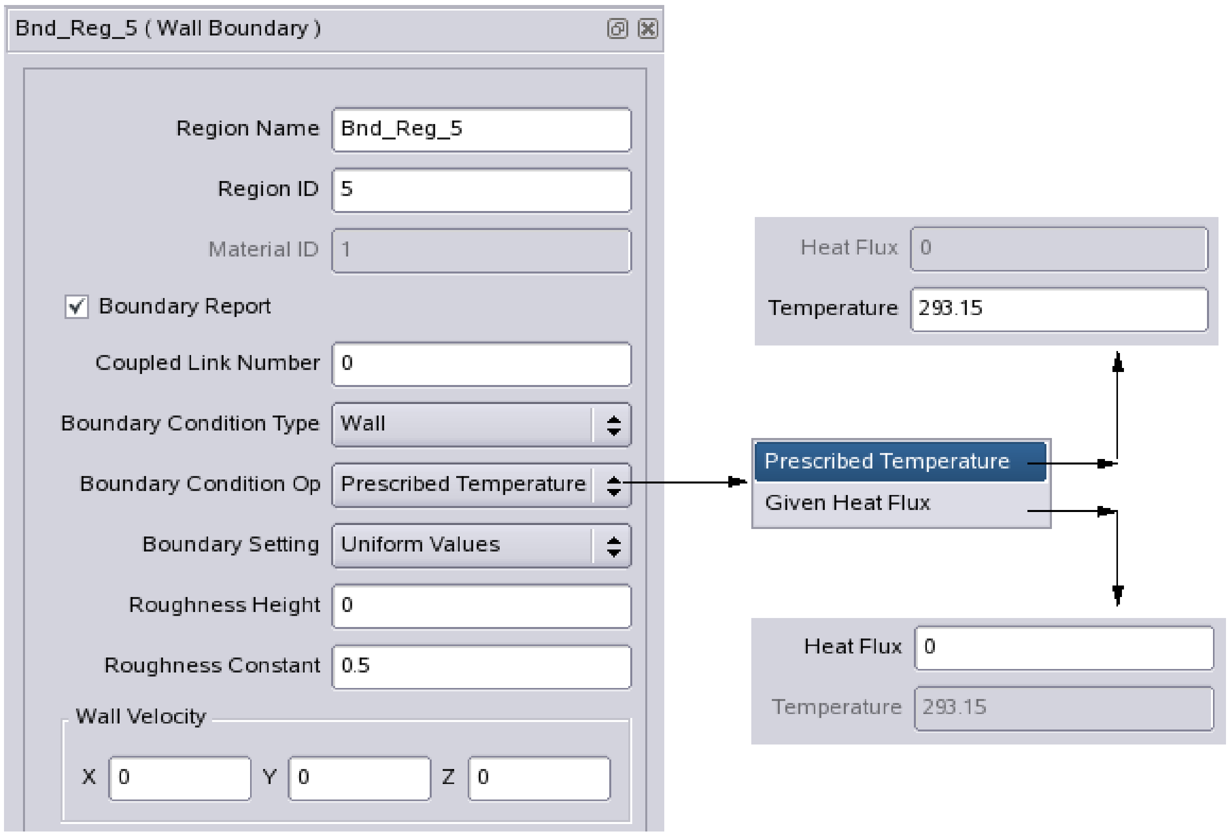
In the Solver Setup Tree left-click on a boundary region to get the boundary setup panel. Under Boundary Condition Type select the Wall option from the ListBox and the Wall Boundary panel containing wall boundary settings is displayed as in Figure 13.9.

The Cartesian velocity components for wall motion can be specified in the Wall Velocity section where zero values for all velocity components indicate a non-moving wall. In case of modelling a turbulent flow where the wall roughness effects are significant the user should provide values for the Roughness Height and Roughness Constant . These values are used in the wall function approach, modified to take into account the roughness effects.

13.8.2 Wall thermal conditions

When solving the energy equation the user can specify one of the following thermal conditions:

- prescribed wall temperature,
- prescribed wall heat flux,
- external convective heat transfer,
- external radiation heat transfer and
- combination of external convective and radiation heat transfer.


Bnd_Reg_5 (Wall Boundary)

Region Name	Bnd_Reg_5
Region ID	5
Material ID	1
✓ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Wall
Boundary Condition Op	Prescribed Temperature
Boundary Setting	Uniform Values
Roughness Height	0
Roughness Constant	0.5

Wall Velocity

X 0 Y 0 Z 0

Prescribed Temperature
Given Heat Flux

Heat Flux	0
Temperature	293.15

Heat Flux	0
Temperature	293.15

Figure 13.9: Setting up a Wall Boundary type in *R–Desk*

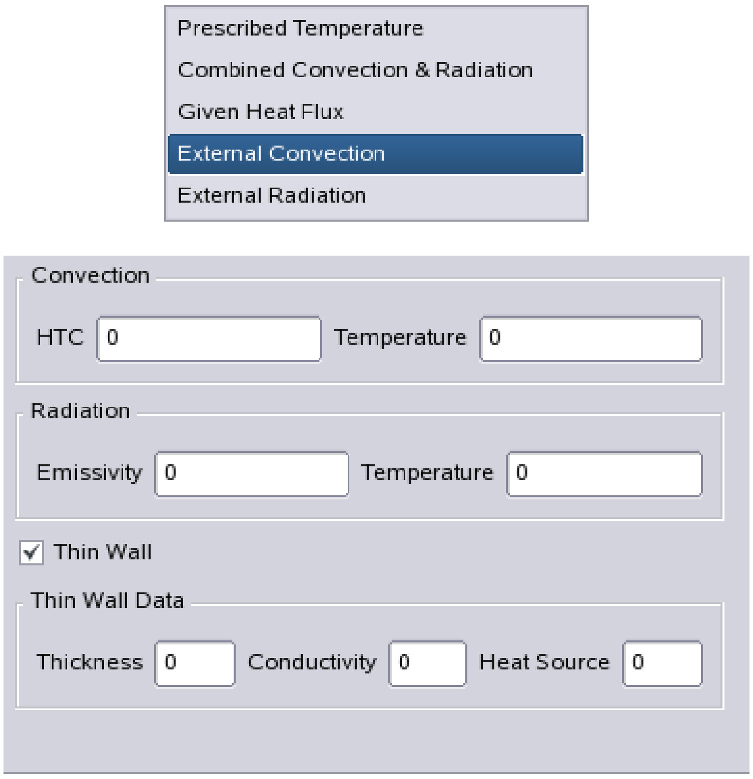
These conditions will be displayed as shown in Figure 13.10(top) after selecting the Thermal Condition Op ListBox.

Each selected wall thermal condition will be accompanied with corresponding LineEdit boxes where the user need to enter required variable values. Figure 13.9(right) illustrates the inputs for the Prescribed Temperature and Given Heat Flux . An adiabatic wall is defined by setting a zero heat flux value. In case of external convection and/or external radiation the required input variables are shown in Figure 13.10(bottom). For external Convection the values of heat transfer coefficients HTC and external Temperature should be specified. If external Radiation heat transfer has been enabled the external Emissivity and external radiation Temperature need to be provided.

13.8.3 Thin–Wall Model

The default wall thickness is assumed to be zero. However, the wall thickness can be taken into account by using the *thin–wall* model which can be used in conjunction with any of the above wall thermal conditions. If the thin–wall model is employed then the above thermal conditions are applicable to the external (outer) side of a thin wall while the internal (inner) side is in the contact with either fluid or solid material domain.

To enable the thin–wall model the Thin Wall box should be checked. Then the following Thin



HTC 0 Temperature 0

Emissivity 0 Temperature 0

[✓] Thin Wall

Thickness 0 Conductivity 0 Heat Source 0

Figure 13.10: Setting up wall thermal conditions in *R–Desk* : drop–down list of all conditions (top) and input variables for the Combined Convection & Radiation in conjunction with Thin Wall (bottom).

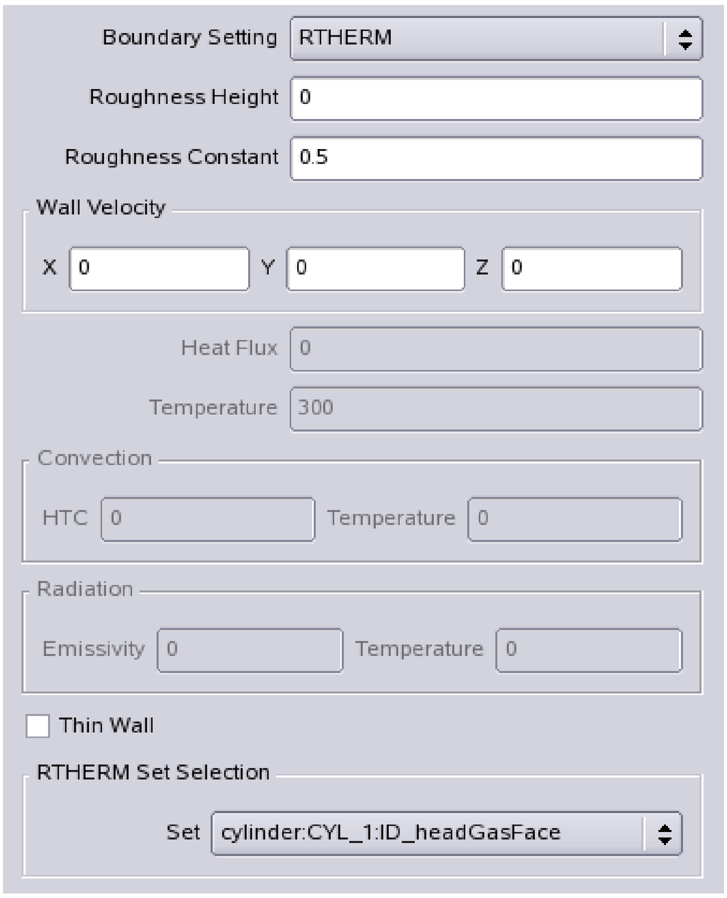
Wall Data should be entered:

- Thickness : this is the wall thickness in $[m]$.
- Conductivity : thermal conductivity of the wall material, $[W/mK]$.
- Heat Source : this is a heat generation rate in $[W/m^3]$.

Note that the thin–wall model will be inactive if its thickness is set to zero.

13.8.4 Setting up thermal conditions via *R–Therm* module

The user can import wall thermal conditions from a RTH–file (Ricardo THermal) when the Boundary Setting option is set to RTHERM , Figure 13.11. The thermal condition RTH–file must be created in advance by using *R–Therm* module. This file contains thermal condition data for various engine components (wall boundaries) which are represented by Ricardo's SFE (Standard Finite Element) sets as shown in Table 13.1. Note that the full RTH set names got the prefix "FE:". Within VECTIS-MAX solver a thermal condition type for each wall is assigned according to the 'Solver Name' types which are also given in Table 13.1.



Boundary Setting	RTHERM
Roughness Height	0
Roughness Constant	0.5

Wall Velocity

X	0
Y	0
Z	0

Heat Flux	0
Temperature	300

Convection

HTC	0
Temperature	0

Radiation

Emissivity	0
Temperature	0

☐ Thin Wall

RTHERM Set Selection

Set	cylinder:CYL_1:ID_headGasFace

Figure 13.11: Using the RTHERM Boundary Setting option to associate the wall boundary with the RTHERM Set Selection .

Engine Component	RTH Base Set Name	Solver Name	Wall Condition
Liner	cylinder:CYL_x:ID_bore	cylinder	variable HF
Flame Face	cylinder:CYL_x:ID_headGasFace	cylinder	variable HF
Inlet Port	port:CYL_x:VAL_Iy	port	uniform EHT
Exhaust Port	port:CYL_x:VAL_Ey	port	uniform EHT
Inlet Valve Contact	valveSeat:CYL_x:VAL_Iy	valveSeat	uniform HF
Exhaust Valve Contact	valveSeat:CYL_x:VAL_Ey	valveSeat	uniform HF

Table 13.1: Description of engine components in the RTH–file and in the solver: x–cylinder number, y–port/valve number, HF–heat flux, EHT–external heat transfer.

R–Therm module defines the engine component geometry in terms of a regular, relatively coarse (finite element) mesh. Thermal data fields (heat flux, external heat transfer coefficient and temperature) are then generated based on the Ricardo's empirical data for each component. These fields are described by thermal values at nodes of the *R–Therm* finite element mesh. As the VECTIS-MAX boundary mesh (describing engine components which are *identical* to those used in *R–Therm* module) is unstructured (irregular) the solver thermal boundary values (defined at boundary face centres) are generated from the *R–Therm* nodal values using a linear interpolation method. The interpolation is required only if the thermal field is not uniform as it is a case for cylinder liners and flame faces.

To perform an interpolation or assign uniform thermal values at solver's wall boundaries the user has to associate the wall boundary marked with the RTHERM setting flag with an *R–Therm* set from the RTH-file. If the user sets a new project and has already created a Domain Structure, Figure 7.2, the Set list of the RTHERM Set Selection shown in Figure 13.11 will be initially empty. After selecting the RTHERM option for the first time within a new project, the Solver Setup Tree will be refreshed and show a new RTHERM node within Global Domain sub–tree, Figure 13.12.

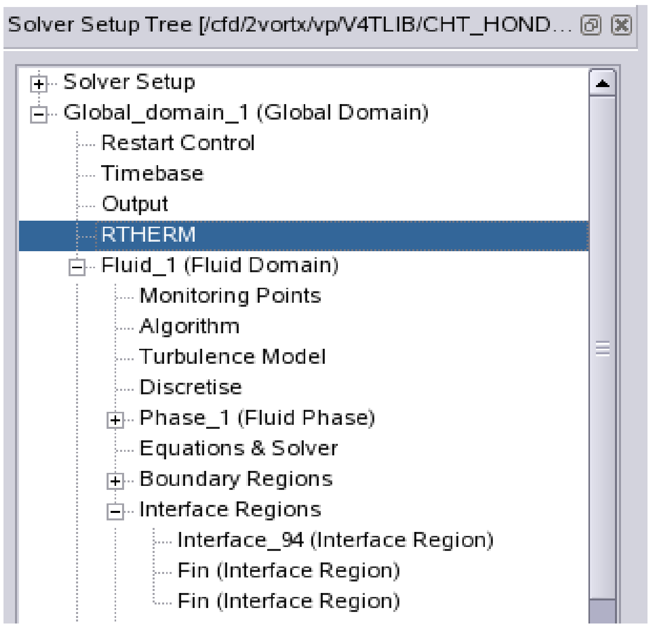
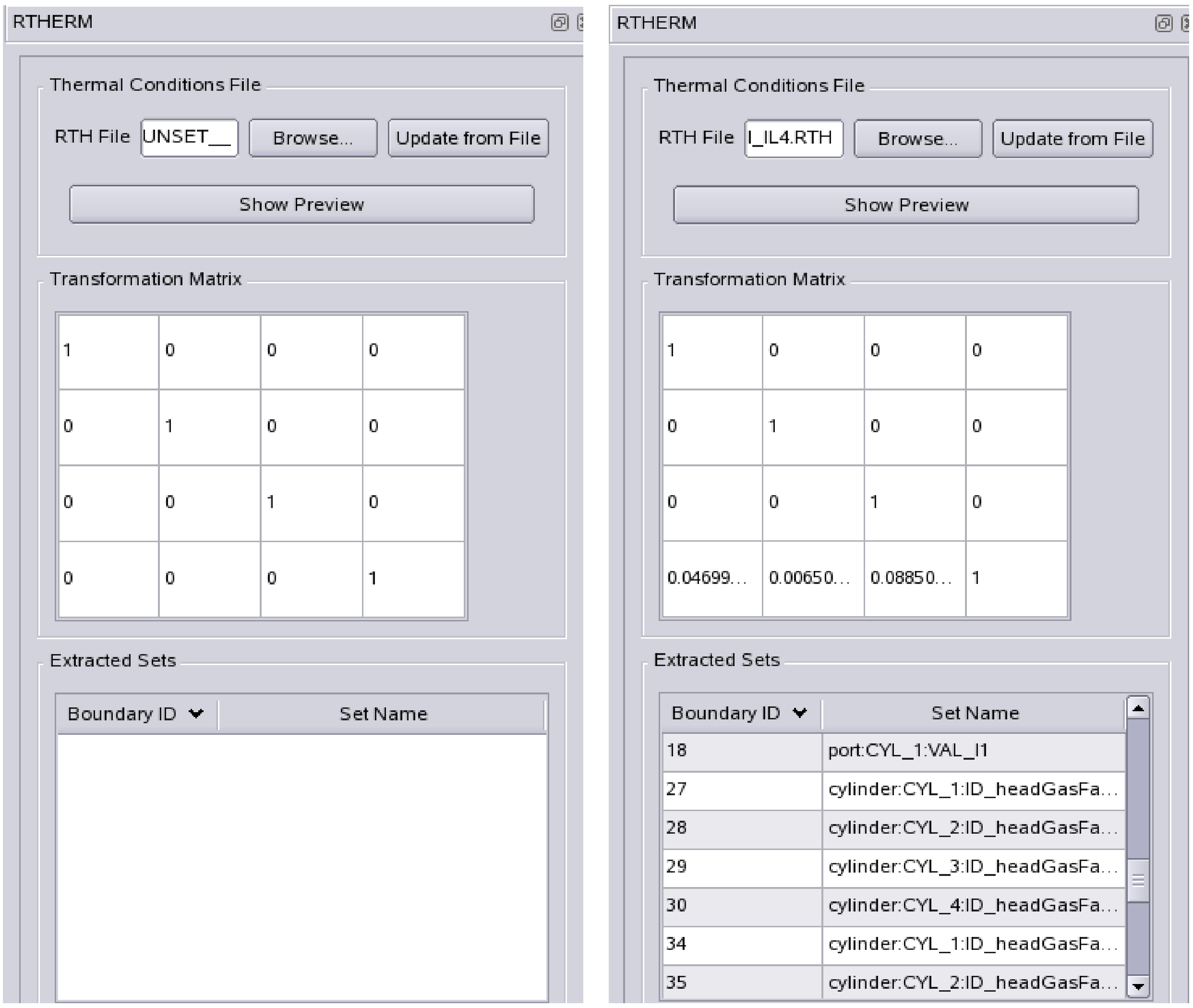


Figure 13.12: RTHERM node as a part of the Solver Setup Tree .

After selection of the RTHERM node an "empty" RTHERM panel will be displayed as shown in Figure 13.13 (left). Assuming that a file containing thermal conditions has been created earlier using *R–Therm* module, the user should be able to Browse and load the required RTH File . Note that a "RTH file" has extension ".RTH".

The Show Preview can be used to display *R–Therm* geometry or mesh. As the solver mesh might be created within a different Cartesian coordinate system than the *R–Therm* mesh, Transformation Matrix need to be edited. This matrix is then used to transform VECTIS-MAX coordinates, more precisely boundary face centres, into *R–Therm* coordinates. In order to obtain the correct transformation matrix the user can preview both *R–Therm* sets and solver boundaries and establish required coordinate system translation or rotation. For this, *R–Desk* distance and angle measuring tools can be used. After loading an RTH–file and creating a transformation matrix all Set Names will appear in the drop–down list of Extracted Sets , Figure 13.13 (right). At this stage, Boundary ID corresponding to each Set Name will not be assigned.

In the next step, the user can revisit the first considered wall boundary. This time RTHERM Set



RTHERM

Thermal Conditions File

RTH File: UNSET__

Transformation Matrix

1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Extracted Sets

Boundary ID	Set Name

RTHERM

Thermal Conditions File

RTH File: I_IL4.RTH

Transformation Matrix

1	0	0	0
0	1	0	0
0	0	1	0
0.04699...	0.00650...	0.08850...	1

Extracted Sets

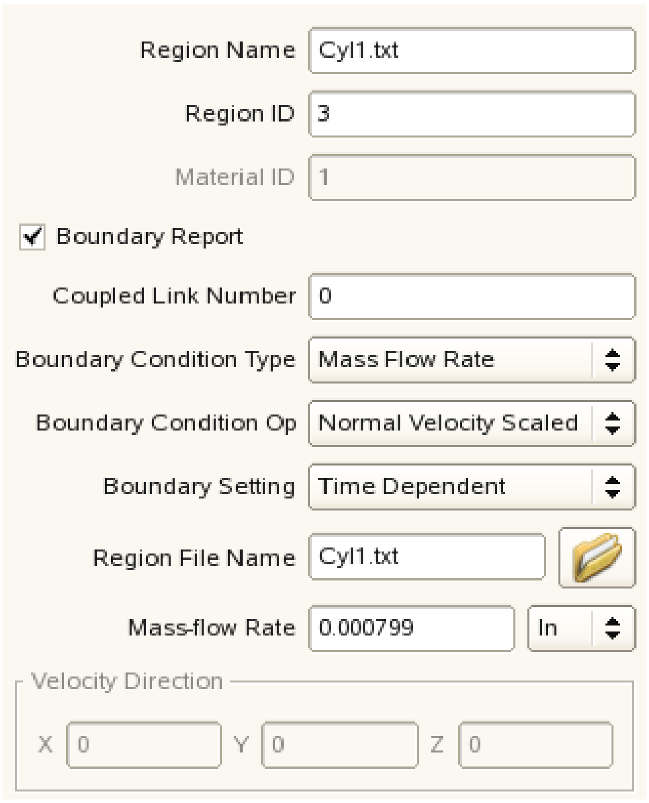
Boundary ID	Set Name
18	port:CYL_1:VAL_I1
27	cylinder:CYL_1:ID_headGasFa...
28	cylinder:CYL_2:ID_headGasFa...
29	cylinder:CYL_3:ID_headGasFa...
30	cylinder:CYL_4:ID_headGasFa...
34	cylinder:CYL_1:ID_headGasFa...
35	cylinder:CYL_2:ID_headGasFa...

Figure 13.13: RTHERM panel in *R–Desk* : after opening (left) and after editing (right).

Selection list will contain all *R–Therm* sets imported from the RTH–file. After selecting a correct *R–Therm* set to be associated with the current wall boundary, the Region ID number will be assigned to the selected *R–Therm* set and will be visible in a drop–down list of Extracted Sets in the RTHERM panel, see Figure 13.11 and Figure 13.13 (right), respectively. In a similar way, other wall boundaries can be associated with the corresponding *R–Therm* sets.

13.9 Setting Time–Dependent Boundary Conditions

Unsteady simulations can be performed in conjunction with time-dependent boundary values. The specification of time-dependent boundary variables is currently provided via ASCII files. As Figure 13.14 illustrates, for each boundary region having Boundary Setting set to Time Dependent option, a separate file should be prepared. The corresponding file name need to be entered into Region File name line box.



Region Name	Cyl1.txt
Region ID	3
Material ID	1
✓ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Mass Flow Rate
Boundary Condition Op	Normal Velocity Scaled
Boundary Setting	Time Dependent
Region File Name	Cyl1.txt
Mass-flow Rate	0.000799 In

Velocity Direction

X	0
Y	0
Z	0

Figure 13.14: *R–Desk* boundary condition panel for time–dependent boundary data

File names are arbitrary and their format follows a VECTIS3 practise. Thus, there are 12 entries for each time-step and these entries can be specified using one or two lines. The meaning of entries is as follows:

1. Time in units of the specified time base
2. Region mass flow rate $[kg/s]$.
3. The boundary region area (not used currently).
4. Region mean temperature, $[K]$.
5. Region mean absolute pressure, $[Pa]$.
6. Region mean turbulent intensity.
7. Region mean turbulent length scale, $[m]$.
8. Region mean value of a passive scalar mass fraction.
9. Entries 9 *to* 12: Region mean values of wave species mass fractions, i.e. mass fractions for air, vapour, burnt air and burnt fuel are specified while the value of liquid fuel is omitted (assumed zero).

In order to perform an unsteady run with time-dependent boundary data the global control variable Time Dependent Boundary Conditions from the Timebase panel need to be activated (ticked on). If it is not activated any time-dependent settings for individual boundary regions will be disabled.

If the real simulation solver time (in seconds) becomes larger than the ending time specified within a boundary region file, the run will switch to time-dependent data corresponding to the start time specified in the region file.

Apart from the time-dependent boundary conditions (supplied via boundary region files), realistic initial velocity, pressure, temperature, turbulence and other variable fields are crucial in obtaining the expected simulation results. Thus it is user's responsibility to provide the initial fields. In the absence of the realistic initial fields, a simple approach could be to obtain the initial fields by performing a steady run using boundary conditions which are those corresponding to the start time as specified within each boundary region data file.

The current contents of boundary region files restrict the specification of time-dependent data to the single-phase flow where the phase can be defined as a single-component or as mixture of the wave species, see Setting a phase. Also, the boundary region types are supposed to be either mass flow or pressure boundaries.

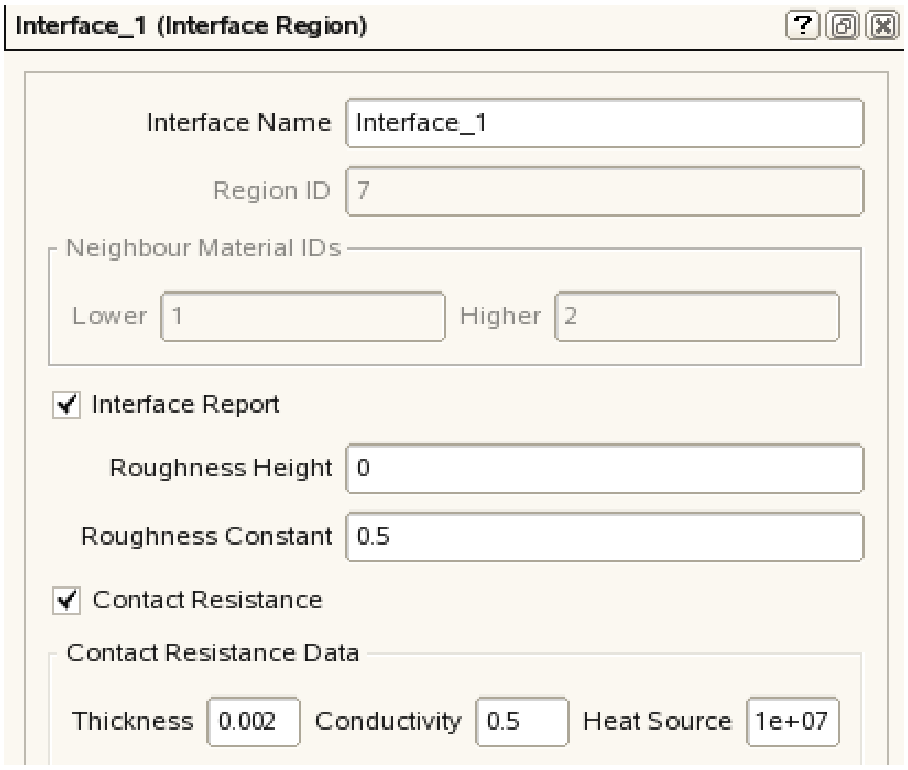
13.10 Setting Up Interface Conditions

For fluid flow equations, with exception of the energy, an interface condition type is the same as the wall boundary type. Thus the interface condition setup is similar to the Basic wall setup.

In order to open the Interface Region panel an Interface Region should be selected (mouse left–click) from the solver Interface Regions sub–tree, see for example Figure 13.12. A typical Interface Region panel is depicted in Figure 13.15. Here, the following interface attributes can be changed:

- ☐ Interface name : a meaningful name should be entered.
- ☐ Interface Report : if an ASCII file report is wanted then this box should be checked. By default, the interface report is written to the SDF binary "rep" file 'projectname.rep_runnumber' where interface variable names in the file contain ':INTF' string. The ASCII file will be named as 'projectname.intf_runnumber'. For more details see File Output section.
- ☐ For turbulent flow simulations, where the wall roughness effects are not negligible, the user should provide values for the Roughness Height in $[m]$ and Roughness Constant
- ☐ Contact Resistance : Similarly to the Thin–Wall Model, the thermal resistance caused by the thickness of a thin wall placed along an interface can be taken into account via thin–interface model.

To enable the thin–interface model the Contact Resistance box should be checked. The Contact Resistance Data group box will appear, and the user should enter the following data:



Interface Name	Interface_1
Region ID	7

Neighbour Material IDs

Lower	1	Higher	2

✓ Interface Report

Roughness Height	0
Roughness Constant	0.5

✓ Contact Resistance

Contact Resistance Data

Thickness	0.002	Conductivity	0.5	Heat Source	1e+07

Figure 13.15: Interface Region panel in *R–Desk* .

- □ Thickness : this is the interface wall thickness in $[m]$.
- □ Conductivity : thermal conductivity of the interface wall material, $[W/mK]$.
- □ Heat Source : this is a heat generation rate in $[W/m^3]$.

The thin interface wall can be placed along an interface between the fluid and solid material or between solid and solid material. Due to thin interface wall thermal resistance, the interface temperature has dual values. Thus one has to distinguish between the temperature at the lower interface (interface side towards a material with a lower index) and between the temperature at the upper interface (interface side towards a material with a higher index). Note that the thin–wall model will be inactive if its thickness is set to zero.

Other attributes can not be changed – they have been extracted from the non–partitioned grid file during Domain Structure creation and they are used by the VECTIS-MAX solver to check the validity of a grid file and extraction process.

14 NUMERICAL SOLUTION

The numerical solution of the modelling, mean–flow equations is based on the finite volume discretisation and on a SIMPLE–like segregated procedure for the velocity–pressure–density coupling, Patankar [1980]; Karki and Patankar [1989]; Demirdzic et al. [1993]; Przulj and Basara [2002].

The following sections describe:

- □ Finite Volume Discretisation considering the discretisation of a generic transport equation;
- □ Poor Quality Cell Treatment and
- □ Solver Parallelisation.

14.1 Finite Volume Discretisation

The solution domain is discretised with an unstructured numerical grid described by edges of (convex) polyhedra (cells). All solution variables are located at cell centroids. The governing equations can be conveniently expressed in an integral form of the general conservation law. Its semi–discrete form is obtained after the approximation of surface and volume integrals by using the values of integrand that prevail at the face and cell centroids, respectively.

14.1.1 Generic transport equation

Equations presented in section 10.1 can be written in the generic integral form:

$$\underbrace{\frac{d}{dt}\int_V \rho\phi dV}_{Transient} + \underbrace{\sum_{j=1}^{n_f}\int_{A_j} \rho\phi\left(U_k - U_{bk}\right) dA_k}_{Convection} = \underbrace{\sum_{j=1}^{n_f}\int_{A_j} \Gamma_\phi^{kk}\frac{\partial\phi}{\partial x_k} dA_k}_{Diffusion} + \underbrace{\int_V s_\phi^V dV + \sum_{j=1}^{n_f}\int_{A_j} s_{\phi k}^A dA_k}_{Sources} \quad (14.1)$$

Each of the terms in Equation (14.1), ϕ, Γ_ϕ^{kk}, $S_{\phi k}^A$ and S_ϕ^V are given appropriate values according to the type of the transport equation being solver for.

Similarly the continuity equation is written in the following integral form:

$$\frac{d}{dt}\int_V \rho dV + \sum_{j=1}^{n_f}\int_{A_j} \rho\left(U_k - U_{bk}\right) dA_k = 0 \tag{14.2}$$

In order to arrive at a set of closed set of modelling equations, the space conservation law which is needed for problems with moving boundaries. In the integral form this equation is written as:

$$\frac{d}{dt}\int_V dV - \sum_{j=1}^{n_f}\int_{A_j} U_{bk} dA_k = 0 \tag{14.3}$$

An analytical solution of the governing equations casted in the integral form does not exist except in the most simplistic form. Most real world problems require the solution of the governing equations presented. A numerical method is therefore used to convert the system of governing equations into a system of algebraic equations that can be solved for. This process of conversion is known as discretisation. A number of discretisation methods exist and the VECTIS-MAX is build around the Cell-Centred discretisation method and is explained in this chapter.

Considering a non-moving control volume, in Figure 14.1, the second order accurate approximation of the generic transport equation (14.1) leads to the following balance equation, written for the cell P with the volume V:

$$\frac{d}{dt}\left(\rho V\phi\right)_P + \sum_{j=1}^{n_f} C_j = \sum_{j=1}^{n_f} D_j + \sum_{j=1}^{n_f}\left(s_{\phi k}^{A} A_k\right)_j + \left(s_{\phi}^{V}\right)_P V_P \tag{14.4}$$

where convection and diffusion fluxes (C_j and D_j) are defined as:

$$C_j = \dot{m}_j \phi_j \ , D_j = \left(\Gamma_\phi \nabla\phi \cdot \vec{A}\right)_j \tag{14.5}$$

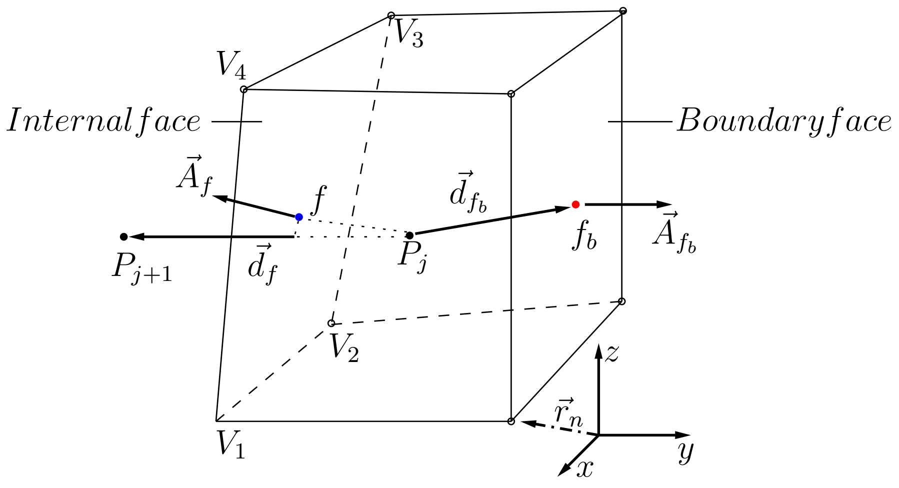


Figure 14.1: Control volume and notation

In the above equation, ϕ is a flow variable and Γ_ϕ represents its (isotropic) diffusion coefficient. The outward face area vector is denoted as $\vec{A} = A_k \vec{i}_k$, where $\vec{i}_k$ is the Cartesian unit vector. The convective and diffusion fluxes through the face j are C_j and D_j, respectively; the source terms s_ϕ^V and s_ϕ^A are related to the cell volume and cell face, respectively; n_f is the number of cell-faces. Mass fluxes $\dot{m}_j$ are defined as:

$$\dot{m}_j = \rho_j \vec{U}_j^* \cdot \vec{A}_j \tag{14.6}$$

and they must satisfy the integral mass balance over each cell:

$$\frac{d}{dt}(\rho V)_P + \sum_{j=1}^{n_f} \dot{m}_j = 0 \tag{14.7}$$

14.1.2 Time Integration

Time integration is needed in unsteady fluid flow problems. The semi-discrete general transport equation (14.4) is written in the form of a first order ordinary differential equation:

$$\frac{d(\rho V \phi)_P}{dt} + \Phi(\phi, t) = 0 \tag{14.8}$$

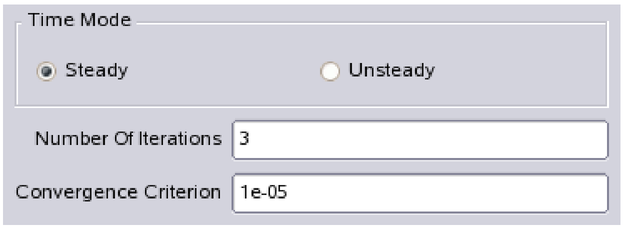
where the term Φ represents the fluxes and sources as:

$$\Phi = \sum_{j=1}^{n_f} C_j - \sum_{j=1}^{n_f} D_j - \sum_{j=1}^{n_f} \left(s_{\phi k}^A A_k\right)_j - \left(s_\phi^V\right)_P V_P \tag{14.9}$$

Equation (14.8) is integrated over each time step δt which advances the solution step by step in time. Then this equation is discretised according to the implicit scheme. There are 2 implicit schemes available: *first order accurate Euler scheme* and *second order accurate three time level scheme*. Both of these schemes are discussed in further in Transient Term section.

14.1.2.1 Setting up a steady or unsteady flow

Using *R–Desk* left-click on Global Domain node in the Solver Setup Tree and the Global Domain panel is displayed to the right. Now parameters for steady and unsteady flow can be set up by selecting the appropriate radio button as in Figure 14.2. In case of steady state simulation, Number of Iterations and Convergence Criterion can be specified. When the flow is unsteady then the user can select the Time Scheme using the ListBox to either be 1st Order Euler or 2nd Order Implicit (three time level scheme). Next the user must select the Time Base , which can be one of Seconds , Non-dimensional , 2-stroke and 4-stroke . For the Non-dimensional case, the Cycle Time , Number of steps per cycle , Start Time and End Time must be provided. When using the 2-Stroke or 4-Stroke option, the Time Step , Start Time , End Time and Engine Speed are required (in the units indicated, e.g. degrees and rev/min). In addition other time dependent related parameters can also be specified such as: Time Dependent Boundary Conditions , Number of Steps , Time step , Maximum Number of Outer Iterations and Convergence Criterion .

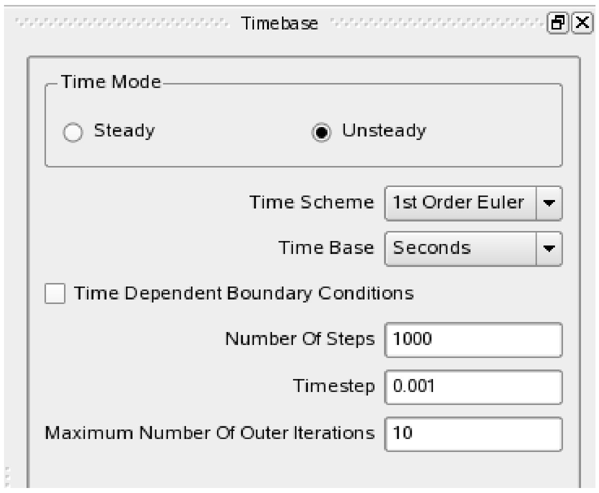


Time Mode

[✓ Steady] Unsteady

Number Of Iterations 3

Convergence Criterion 1e-05



Timebase

Time Mode

Steady [✓ Unsteady]

Time Scheme 1st Order Euler

Time Base Seconds

☐ Time Dependent Boundary Conditions

Number Of Steps 1000

Timestep 0.001

Maximum Number Of Outer Iterations 10

Figure 14.2: *R–Desk* setup for steady and unsteady simulations.

14.1.3 Interpolative schemes for cell–face values

Obviously, the cell–face values and gradients are important ingredients of the discretisation procedure. They can be calculated using linear interpolation (CDS) between the values at adjacent cells P and P_j (Figure 14.3):

$$\phi_j = f_j \phi_P + \left(1 - f_j\right) \phi_{P_j} \tag{14.10}$$

where the interpolation factor f_j is given as

$$f_j = \frac{|\vec{r}_{P_j} - \vec{r}_j|}{|\vec{r}_{P_j} - \vec{r}_j| + |\vec{r}_j - \vec{r}_P|} \tag{14.11}$$

and $\vec{r}$ denotes the position vector. Since CDS is not always appropriate for the convective term, a simple and efficient approach is to blend the CDS scheme with a small amount of the UDS scheme Demirdzic and Muzaferija [1995],Ferziger and Peric [1997]. The blending is done by introducing a blending factor $0 \leq \gamma_\phi \leq 1$ and can be arranged as:

$$\phi_j = \phi_j^{UDS} + \gamma_\phi f_j^* \frac{\dot{m}_j}{|\dot{m}_j|} \left(\phi_{P_j} - \phi_P\right) \tag{14.12}$$

$$f_j^* = \begin{cases} f_j & \text{if} \quad \dot{m}_j < 0 \\ 1 - f_j & \text{if} \quad \dot{m}_j \geq 0 \end{cases} \tag{14.13}$$

with f_j^* being the 'flow orientated' interpolation factor. The blending factor depends on the grid resolution and for sufficiently fine grids the values close to one can be used. The alternative is to adopt the high resolution bounded scheme which were discussed in Przulj and Basara [2001] but also covered here.

14.1.3.1 Boundedness criteria

Let us consider transport of a scalar ϕ along the local coordinate ξ, which passes through the upstream (U), central (C) and downstream (D) computational nodes, Figure 14.3. The actual labelling of these nodes depends on the velocity direction, i.e. on the sign of mass flow rate $\dot{m}_j$. We require that the cell face value ϕ_j fulfils the following two conditions:

- □ it is bounded by the neighbouring cell values ϕ_C and ϕ_D and
- □ enforces monotonicity of the local function $\phi(\xi)$ which passes through the points ϕ_U and ϕ_C.

The first condition represents the interpolative boundedness Gaskell and Lau [1988] and can be generalised as:

$$0 \le \frac{\phi_j - \phi_C}{\phi_D - \phi_C} \le 1 \ \text{ or } \ 0 \le \frac{\widetilde{\phi}_j - \widetilde{\phi}_C}{1 - \widetilde{\phi}_C} \le 1 \tag{14.14}$$

The second, monotonicity condition, reads:

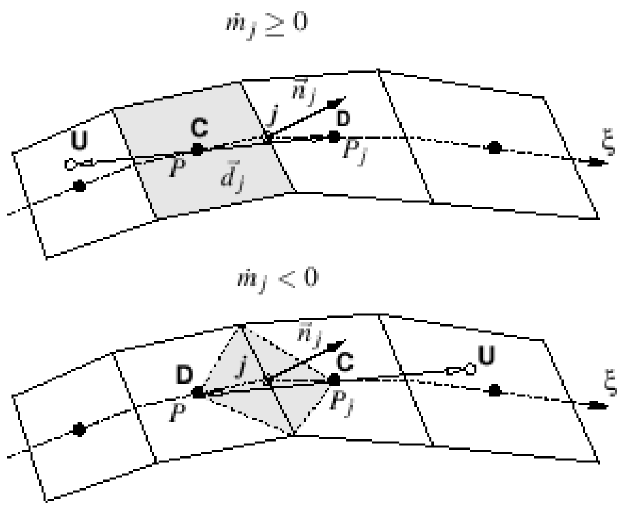


Figure 14.3: Definition of upstream, central and downstream nodes.

$$\frac{\phi_j - \phi_C}{\phi_C - \phi_U} \ge 0 \ \text{ or } \ \frac{\widetilde{\phi}_j - \widetilde{\phi}_C}{\widetilde{\phi}_C} \ge 0 \tag{14.15}$$

In the above equations, the normalised variable Leonard [1988] $\widetilde{\phi}$ is introduced:

$$\widetilde{\phi} = \frac{\phi - \phi_U}{\phi_D - \phi_U} \tag{14.16}$$

giving $\widetilde{\phi}_U = 0$ and $\widetilde{\phi}_D = 1$. It is not difficult to prove that the inequalities (14.14) and (14.15) are simultaneously satisfied if the following relations hold:

$$\begin{array}{c} \widetilde{\phi}_j \le 1 \ \ \& \ \ \widetilde{\phi}_j \ge \widetilde{\phi}_C \ \ \text{if} \ \ 0 < \widetilde{\phi}_C < 1 \\ \widetilde{\phi}_j = \widetilde{\phi}_C \ \ \text{if} \ \ \widetilde{\phi}_C \le 0 \ \ \text{or} \ \ \widetilde{\phi}_C \ge 1 \end{array} \tag{14.17}$$

These are CBC constraints derived earlier by Gaskell and Lau Gaskell and Lau [1988]. In the case of the monotonic profile, characterised by $(\phi_D - \phi_C)/(\phi_C - \phi_U) \ge 1$, we might prefer to bound the cell face value by the extrapolated downstream value $\phi_D^e = 2\phi_C - \phi_U$ rather than by the actual value ϕ_D. If this constraint, expressed as:

$$\frac{\phi_j - \phi_C}{\phi_C - \phi_U} \leq 1 \quad \text{or} \quad \frac{\widetilde{\phi}_j - \widetilde{\phi}_C}{\widetilde{\phi}_C} \leq 1 \tag{14.18}$$

is added to the previous inequalities (14.14) and (14.15), the TVD criteria are obtained:

$$\begin{array}{c} \widetilde{\phi}_j \leq 1 \;\;\&\;\; \widetilde{\phi}_j \leq 2\widetilde{\phi}_C \;\;\&\;\; \widetilde{\phi}_j \geq \widetilde{\phi}_C \;\; \text{if} \;\; 0 < \widetilde{\phi}_C < 1 \\ \widetilde{\phi}_j = \widetilde{\phi}_C \;\; \text{if} \;\; \widetilde{\phi}_C \leq 0 \;\; \text{or} \;\; \widetilde{\phi}_C \geq 1 \; . \end{array} \tag{14.19}$$

For arbitrary unstructured grids the upstream node is not known. However, an imaginary upstream cell can be defined in such a way that the vector identity $\vec{CU} = -\vec{CD}$ is satisfied and that an imaginary face between C and U is placed at the same distance from C as the considered face j, Figure 14.3. In this case, the centred difference $(\phi_D - \phi_U)$ is simply $\nabla\phi_C \cdot 2\vec{d}_j$, which yields the reconstructed value ϕ_U^* as:

$$\phi_U^* = \phi_D - \nabla\phi_C \cdot 2\vec{d}_j \tag{14.20}$$

The above reconstructed value need to be bounded by the values at neighbouring cells which surround the upstream node. This can be done by checking ϕ_U^* against the minimum and maximum values of ϕ over the cells that share each vertex of the central cell C:

$$\phi_U = max\left[\phi_{min}^{neighbours}, min\left(\phi_U^*, \phi_{max}^{neigbours}\right)\right] \tag{14.21}$$

Our experience has shown that the computationally less expensive procedure, which involves only the neighbours that share faces of the central cell, produces satisfactory results and this procedure is used in this work.

The TVD and CBC conditions are illustrated in Figure 14.4, in the form of a Normalised Variable Diagram (NVD). It follows that the cell–face values $\widetilde{\phi}_j$ should lie within the shaded areas in the monotonic range $0 < \widetilde{\phi}_C < 1$, and on the line $\widetilde{\phi}_j = \widetilde{\phi}_C$ outside the monotonic range. Obviously, the convective schemes with linear characteristics in the NVD diagram (CDS, LUDS, QUICK) may violate boundedness criteria and some form of a non–linear or piecewise linear scheme need to be used. It is useful for many reasons to express a general function $\widetilde{\phi}_j = f(\widetilde{\phi}_C)$ in a form that involves the flux limiter Sweby [1984] $0 \leq \varphi_j(r_j) \leq 1$:

$$\widetilde{\phi}_j = \widetilde{\phi}_C + \varphi_j\left(1 - \widetilde{\phi}_C\right) \tag{14.22}$$

The flux limiter itself is a function of the argument r_j, which is defined as:

$$r_j = \frac{\widetilde{\phi}_C}{1 - \widetilde{\phi}_C} = \frac{\phi_C - \phi_U}{\phi_D - \phi_C} \tag{14.23}$$

In terms of unnormalised variables Equation (14.22) becomes:

$$\phi_j = \phi_j^{UDS} + \varphi_j \frac{\dot{m}_j}{|\dot{m}_j|}\left(\phi_{P_j} - \phi_P\right) \tag{14.24}$$

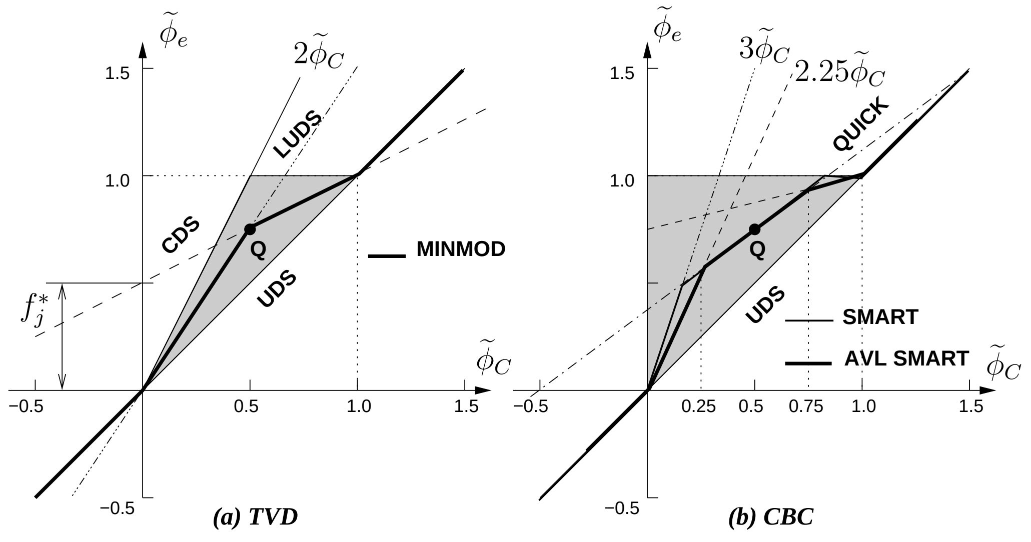


Figure 14.4: NVD diagram: TVD and CBC criteria and characteristics of several convective schemes.

As Figure 14.4 shows, the basic schemes such as CDS, LUDS and QUICK are linear and therefore not always bounded. A general form of these schemes applicable to the uniform grids is derived in Gaskell and Lau [1988]. Taking into account the grid spacing for the situation shown in Figure 14.3, the unbounded limiter can be obtained as:

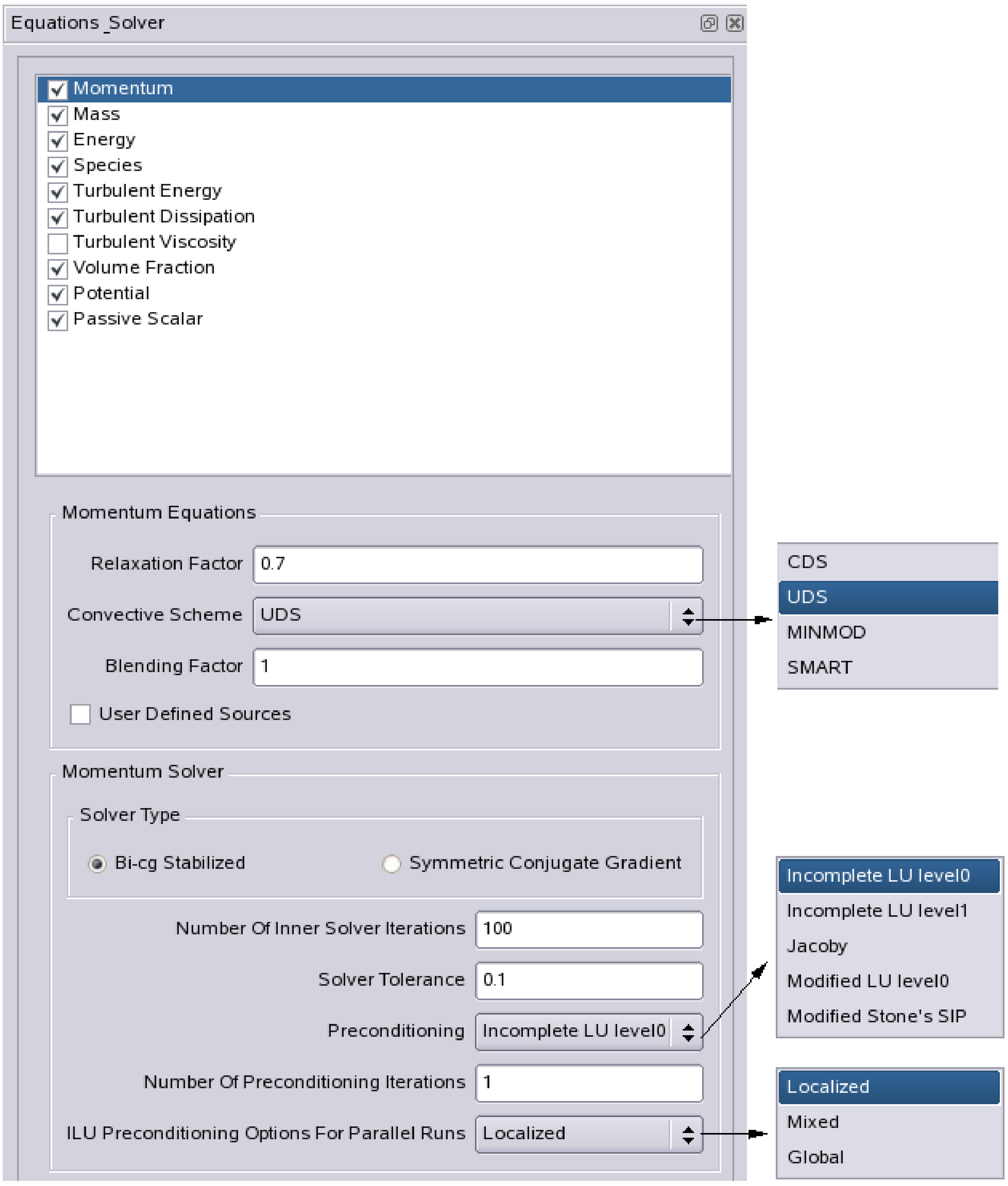
$$\varphi_j = \left(g_D - \alpha_j\right) + \left(g_U + \alpha_j\right) r_j \tag{14.25}$$

$$g_D = \frac{1}{2} f_j^* \left(1 + f_j^*\right) \ , \ g_U = \frac{1}{2} f_j^* \left(1 - f_j^*\right) \tag{14.26}$$

The parameter α defines a family of the second and third–order accurate schemes. For example, $\alpha = 0$ gives the QUICK scheme and $\alpha = 0.5 f_j^* (1 + f_j^*)$ gives the LUDS scheme.

14.1.3.2 Setting up the convective scheme

A number of convective schemes are also available through *R–Desk* GUI setup. Under each Fluid Domain node in the Solver Setup Tree panel, left-click on Equations & Solver node and the Equations_Solver panel is displayed to the right. For each transport equation being solved for the given 'Fluid Domain' one of four available convective schemes can be selected from the ListBox Convective Scheme : UDS , CDS , MINMOD and SMART . The value of the Blending Factor and Relaxation Factor can also be specified. The panel also includes an option for User Defined Sources (see upr_sources). An example for Momentum Equations is given in Figure 14.5. The solver part in Figure 14.5 in discussed in Section 14.2.6.



Equations _Solver

- ✓ Momentum
- ✓ Mass
- ✓ Energy
- ✓ Species
- ✓ Turbulent Energy
- ✓ Turbulent Dissipation
- ☐ Turbulent Viscosity
- ✓ Volume Fraction
- ✓ Potential
- ✓ Passive Scalar

Momentum Equations

Relaxation Factor	0.7
Convective Scheme	UDS
Blending Factor	1

☐ User Defined Sources

Momentum Solver

Solver Type

[✓ Bi-cg Stabilized] Symmetric Conjugate Gradient

Number Of Inner Solver Iterations	100
Solver Tolerance	0.1
Preconditioning	Incomplete LU level0
Number Of Preconditioning Iterations	1
ILU Preconditioning Options For Parallel Runs	Localized

Figure 14.5: *R–Desk* setup for convective schemes and linear solvers

14.1.4 Gradient computation

Two different methods are identified for computing the cell gradient: *Gauss based* and *least-square based*.

Gradient based on the *Gauss' divergence theorem* is calculated as:

$$\nabla\phi_P \approx \frac{1}{V_P}\sum_{j=1}^{n_f}\phi_j\vec{A}_j \tag{14.27}$$

Cell gradient can also be calculated by using *linear least square minimisation method* (Barth

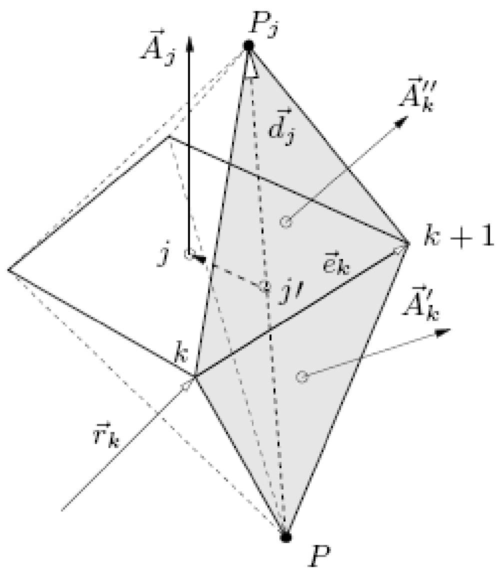


Figure 14.6: Polyhedral control volume around the cell–face j

[1994], Muzaferija [1994]) which minimises the sum of squares of the differences $\left[\phi_{Pj}-\phi(\vec{r}_{Pj})\right]$ assuming the linear distribution between neighbouring nodes P and P_j:

$$\phi(\vec{r}) = \phi_P + \nabla\phi_P\cdot(\vec{r}-\vec{r}_P). \tag{14.28}$$

To compute the face gradient, the Gauss' formula 14.27 can be applied to the control volume constructed around the face j, Figure 14.6. This control volume is made of triangular faces $\vec{A}_k'$ and $\vec{A}_k''$ defined by the face vertices k and $k+1$ (for the last vertex $k+1=1$) and neighbouring nodes P and P_j. If the vertex values are reconstructed from the variable values and gradients at adjacent cells P and P_j, the derivation outlined in Przulj and Basara [2002] leads to the following expression for the face gradient:

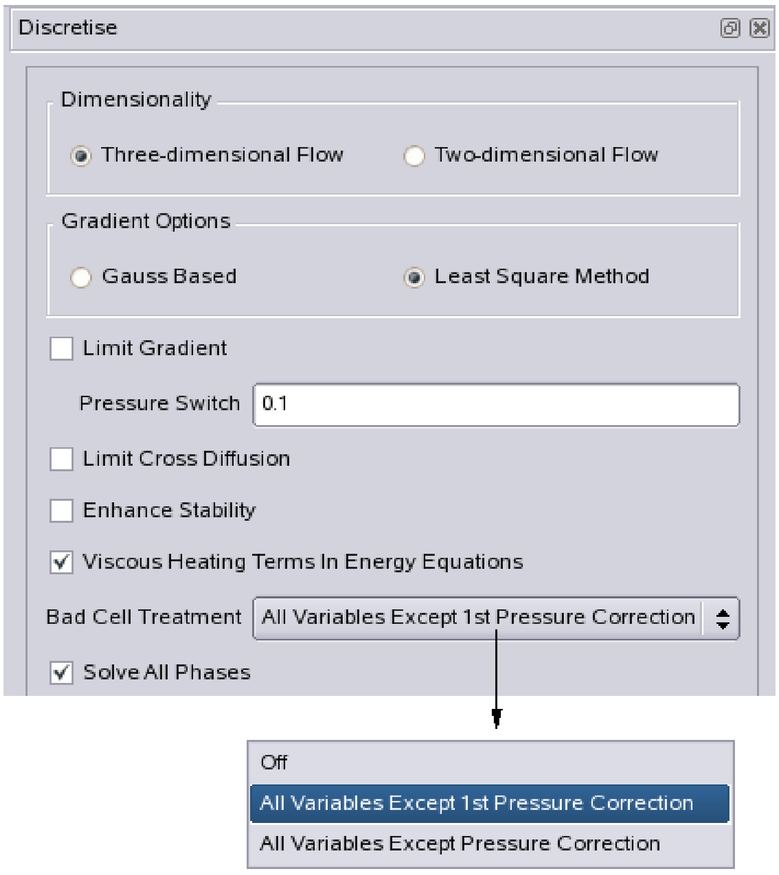
$$\nabla\phi_j \approx \overline{\nabla\phi}_j + \frac{\vec{A}_j}{\vec{A}_j\cdot\vec{d}_j}\left[\left(\phi_{P_j}-\phi_P\right)-\overline{\nabla\phi}_j\cdot\vec{d}_j\right]\ ,\ \overline{\nabla\phi}_j = f_j\nabla\phi_P + (1-f_j)\nabla\phi_{P_j} \tag{14.29}$$

14.1.4.1 Setting up dimensionality and gradient calculation options

>From the Solver Setup Tree under Fluid Domain , left-click on Discretise . This opens up the Discretise panel as seen in Figure 14.7 . The Dimensionality of the problem can be specified.

Selecting the Two-dimensional Flow RadioButton will set the flow to be 2-Dimensional whereas the Three-dimensional Flow will set the flow to be 3-Dimensional.

Cell gradient calculation methods are available for each fluid and solid domains. In the Discretisation Setup panel under Gradient Options two methods are available: Gauss Based and Least Square Method as seen in Figure 14.7. Gradient can be limited by selecting Limit Gradient CheckBox from the same panel. Cross diffusion can also be limited by selecting Limit Cross Diffusion CheckBox.


Discretise

Dimensionality
● Three-dimensional Flow ○ Two-dimensional Flow

Gradient Options
○ Gauss Based ● Least Square Method

☐ Limit Gradient

Pressure Switch: 0.1

☐ Limit Cross Diffusion

☐ Enhance Stability

☑ Viscous Heating Terms In Energy Equations

Bad Cell Treatment: All Variables Except 1st Pressure Correction

☑ Solve All Phases

Off
All Variables Except 1st Pressure Correction
All Variables Except Pressure Correction

Figure 14.7: *R–Desk* setup for 2D or 3D, gradient calculation options etc.

14.1.5 Discretisation of the generic equation

14.1.5.1 Transient term

First, the unsteady term in (14.4) and (14.8) which is mainly used for time-dependent calculation is integrated over each time interval Δt by either the first order accurate Euler or second order accurate three time level scheme, see Ferziger and Peric [1997]. Both schemes are implicit and unconditionally stable. When the first order Euler scheme is applied then the transient term is discretised as:

$$\frac{d}{dt}(\rho V\phi)_P = \frac{(\rho V\phi)_P^{n+1} - (\rho V\phi)_P^n}{\Delta t} \tag{14.30}$$

When the three time level scheme is applied then the transient term is discretised as:

$$\frac{d}{dt}(\rho V\phi)_P = \frac{3(\rho V\phi)_P^{n+1} - 4(\rho V\phi)_P^n + (\rho V\phi)_P^{n-1}}{2\Delta t} \tag{14.31}$$

14.1.5.2 Convection term

The convective flux C_j, Equation (14.4), can be discretised as:

$$C_j = \dot{m}_j \phi_j^{UDS} + \underline{\gamma_\phi |\dot{m}_j| \varphi_j \left(\phi_{P_j} - \phi_P\right)} \tag{14.32}$$

where γ_ϕ is the blending factor between Upwind Differencing Scheme (UDS) and higher order (bounded) schemes and φ_j represents the flux limiter for the selected bounded scheme. The first term in the above equation is treated implicitly, while the second, underlined term, is calculated by using values from the previous iteration step and treated as an additional source term (deferred correction approach Khosla and Rubin [1974]).

14.1.5.3 Diffusion term

With the cell-face gradient defined by Equation (14.29), the diffusion flux D_j in Equation (14.4) becomes:

$$D_j = \underbrace{\overline{\Gamma}_{\phi j} \frac{A_j^2}{\vec{A}_j \cdot \vec{d}_j} \left(\phi_{P_j} - \phi_P\right)}_{D_{j,n}\text{: normal diffusion}} + \underbrace{\overline{\Gamma}_{\phi j} \overline{\nabla\phi}_j \cdot \left(\vec{A}_j - \frac{A_j^2}{\vec{A}_j \cdot \vec{d}_j}\vec{d}_j\right)}_{D_{j,c}\text{: cross–diffusion}} \tag{14.33}$$

The non–orthogonal (cross–diffusion) term, which vanishes on orthogonal grids, is treated in a deferred correction manner.

Limiting cross–diffusion. As Equation (14.33) shows, the diffusion flux consists of the normal $D_{j,n}$ and cross–diffusion $D_{j,c}$ terms. The latter is important as it maintains the second order accuracy of the diffusion discretisation. The amount of the cross–diffusion depends on the *face (skewness) angle* θ_j which is defined as an angle between the face area vector $\vec{A}_j$ and the distance vector $\vec{d}_j = \vec{r}_{P_j} - \vec{r}_P$, i.e. its cosine is given as

$$\cos\theta_j = \vec{A}_j \cdot \vec{d}_j / (|\vec{A}_j||\vec{d}_j|) \tag{14.34}$$

Very large face angles (above 75°; if they are greater than 90° the mesh is in principle invalid) will at least slow down the solution convergence. A simple measure would be to neglect the cross–diffusion above the certain face angle. A better compromise between accuracy and numerical stability is achieved by limiting the cross–diffusion by the normal diffusion. The following expression describes this limiting:

$$D_{j,c}^{lim} = D_{j,c} \min\left(1\ , \gamma^{lim} \frac{|D'_{j,n}|}{|D_{j,c}|}\right) \tag{14.35}$$

where $\gamma^{lim} = 1$ is the cross–diffusion limiting factor. In our experience, the use of the alternative definition for the normal diffusion in the above equation:

$$D'_{j,n} = \overline{\Gamma}_{\phi j}\overline{\nabla\phi}_j \cdot \frac{A_j^2}{\vec{A}_j \cdot \vec{d}_j}\vec{d}_j \tag{14.36}$$

improves convergence properties. The same limiting can be applied to boundary cross–diffusion fluxes.

14.1.5.4 Source term

Surface and volume sources in Equation (14.4) are calculated explicitly using the mid-point rule.

When the volume source term $S_\phi^V = f(\phi)$ depends on ϕ it is linearised implicitly (see Patankar [1980]) as:

$$S_\phi^V = S_\phi^C - S_\phi^P \phi_P \tag{14.37}$$

The constraints $S_\phi^C \geq 0$ and $S_\phi^P \geq 0$ are applied to Equation (14.37). This ensures that diagonal dominance remains positive and also it is increased when the term S_ϕ^P is added.

14.1.6 Discretisation of the energy equation

A peculiarity of the energy Equation (6.15) is that the convective and unsteady terms are written in terms of total enthalpy (energy) whereas the diffusion flux is naturally written in terms of temperature gradient. However, the energy equation should be discretised in terms of the same dependent variable, i.e. either total enthalpy or temperature.

It is straightforward to express the temperature gradient as a function of total enthalpy and other variables but it is more desirable to define boundary conditions in terms of temperature. An effective discretisation technique, which describes both diffusion and boundary conditions in terms of temperature, has been designed by applying the deferred correction approach. To this end, we can write the diffusion flux $D_j(T)$ in terms of temperature according to Equation (14.33) and add with opposite signs two equivalent diffusion terms discretised in terms of total enthalpy:

$$D_j = \left[\overline{\Gamma}_{Hj}\frac{A_j^2}{\vec{A}_j \cdot \vec{d}_j}\left(H_{P_j} - H_P\right)\right]^{(n+1)} + \left[D_j(T) - \overline{\Gamma}_{Hj}\frac{A_j^2}{\vec{A}_j \cdot \vec{d}_j}\left(H_{P_j} - H_P\right)\right]^{(n)} \tag{14.38}$$

where superscripts (n) and $(n+1)$ denote current and new iteration steps, respectively. The first, total enthalpy term is now treated implicitly, while remaining terms having values at the current iteration are treated explicitly.

In case of conjugate heat transfer, however, solving the total enthalpy in terms of temperature has a number of advantages, Murthy and Mathur [1998]. Notably, temperature is continuous across fluid–solid interface which is not the case with total enthalpy. Total enthalpy experiences the jump discontinuity which need to be addressed when calculating the diffusion flux. To use temperature as the primary variable we need to express the convective flux, Equation (14.32), in terms of temperature. In this equation, the second term, which represents contribution of the high–order convective schemes should remain as it is, i.e. defined in terms of total enthalpy. The first term is based on the upwind scheme (UDS) and this term contributes towards coefficients of the discretised equation. Considering an iterative solution procedure, it can be re–arranged as:

$$C_j^{UDS} = \underbrace{\left[\frac{\max\left(-\dot{m}_j, 0\right) H_{P_j}}{T_{P_j}}\right]^{(n)}}_{(a_j^c)_{P_j}} T_{P_j}^{(n+1)} + \underbrace{\left[\frac{\max\left(\dot{m}_j, 0\right) H_P}{T_P}\right]^{(n)}}_{(a_j^c)_P} T_P^{(n+1)} \tag{14.39}$$

Recognisably, $(a_j^c)_{P_j}$ and $(a_j^c)_P$ are convection coefficients associated with values of temperature at neighbouring nodes P_j and P. With the help of the discrete continuity Equation (14.7) and following Patankar [1980], corresponding contributions to the central coefficients can be derived as:

$$(a_P)_P^c = \max\left(-\dot{m}_j, 0\right)\left(\frac{H_P}{T_P}\right)^{(n)}, (a_P)_{P_j}^c = \max\left(\dot{m}_j, 0\right)\left(\frac{H_{P_j}}{T_{P_j}}\right)^{(n)} \tag{14.40}$$

Regarding the unsteady term,

$$\frac{d}{dt}(\rho V H) = \frac{d}{dt}(\rho V c_p T) + \frac{d}{dt}\left[\rho V\left(\frac{U_i U_i}{2} + k\right)\right] \tag{14.41}$$

the first term containing temperature is treated implicitly, while the second term is absorbed explicitly into source. Similar treatment as in the above equation was used by Murthy and Mathur [1998] for both unsteady and convection terms.

14.1.6.1 Diffusion flux at interfaces for conjugate heat transfer

The total enthalpy or energy equation is solved in terms of temperature in a fully implicit manner over the entire solution domain, involving all participating fluid and solid domains. At material interfaces a conformal mesh is required so that these interfaces are topologically the same entities as internal fluid or solid cell–faces. Considering the interface j between two different materials (Figure 14.8, left), both temperature and heat flux continuity must be satisfied, ensuring the overall conservativeness of the discretisation. This means that heat transfer from the cell P to the interface j,

$$(D_j)_P = h_P\left(T_j - T_P\right) + (D_{j,c})_P h_P = \Gamma_P \frac{A_j^2}{\vec{A}_j \cdot \vec{d}_{P,j}}, \ (D_{j,c})_P = \nabla T_P \cdot \left(\Gamma_P \vec{A}_j - h_P \vec{d}_{P,j}\right) \tag{14.42}$$

and from the interface j to the cell N,

$$(D_j)_N = h_N\left(T_N - T_j\right) + (D_{j,c})_N h_N = \Gamma_N \frac{A_j^2}{\vec{A}_j \cdot \vec{d}_{j,N}}, \ (D_{j,c})_N = \nabla T_N \cdot \left(\Gamma_N \vec{A}_j - h_N \vec{d}_{j,N}\right) \tag{14.43}$$

must be in balance:

$$D_j = (D_j)_P = (D_j)_N \tag{14.44}$$

Note that Γ in the above equations represents the thermal conductivity; for fluid materials it is the effective wall conductivity given by Equation (9.40). Eliminating T_j from the last identity we can derive the interface heat flux and temperature as:

$$D_j = h_j (T_N - T_P) + \frac{\widetilde{\Gamma}_j}{\Gamma_P \Gamma_N} \left[(1 - f_j)(D_{j,c})_P + f_j (D_{j,c})_N \right] \ , \ h_j = \widetilde{\Gamma}_j \frac{A_j^2}{\vec{A}_j \cdot \vec{d}_j} \tag{14.45}$$

$$T_j = \frac{\widetilde{\Gamma}_j}{\Gamma_P \Gamma_N} \left\{ \left(f_j \Gamma_P T_P + (1 - f_j) \Gamma_N T_N \right) + f_j (1 - f_j) \frac{\vec{A}_j \cdot \vec{d}_j}{A_j^2} \left[(D_{j,c})_N - (D_{j,c})_P \right] \right\} \tag{14.46}$$

The interface conductivity $\widetilde{\Gamma}_j$ is defined as the harmonic mean of the conductivities at P and N:

$$\widetilde{\Gamma}_j = \frac{\Gamma_P \Gamma_N}{f_j \Gamma_P + (1 - f_j) \Gamma_N} \tag{14.47}$$

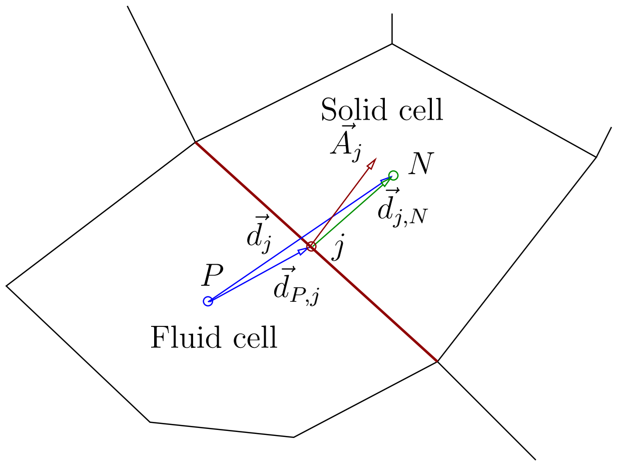


Figure 14.8: Control volumes near fluid–solid interface.

The first, normal diffusion term of the interface flux is included implicitly in the discretised equations for the cells P and N. For the second, cross–diffusion term, an explicit treatment is used. This term can be limited as discussed earlier. The interface temperature T_j is updated after the solution of temperature and then used to compute cell gradients.

14.2 SIMPLE–based solution procedure

The outcome of the discretisation of Equation (14.4) is a set of algebraic equations: one for each control volume and for each transport equation and has the following general form:

$$a_P \phi_P = \sum_{j=1}^{n_i} a_j \phi_{P_j} + S_\phi \, , \tag{14.48}$$

where n_i is the number of internal cell-faces; a_P and a_j are the central and neighbour coefficients, and S_ϕ is the source term.

For this coupled system, a segregated SIMPLE-like solution algorithm, applicable to both incompressible and high speed compressible flows on arbitrary grids, is employed. The SIMPLE algorithm ensures that the calculated flow field, once converged, simultaneously satisfies both the continuity and momentum equations. On non–staggered grids, a special interpolation practice is required for the mass conserving face velocity $\vec{U}_j^*$ in Equation (14.6).

In what follows we consider the pressure correction method which provides velocity, pressure and density coupling for flows at all speeds.

14.2.1 Pressure correction scheme

The SIMPLE algorithm effectively couples the velocity and pressure fields by converting a discrete form of the continuity equation (14.7) into an equation for the pressure correction. The pressure corrections are then used to update the pressure and velocity fields so that the velocity components obtained from the solution of momentum equations satisfy the continuity equation.

14.2.2 Face velocity for mass conservation.

On non-staggered grids, the use of linear interpolation to obtain the pressure and velocities at the cell faces usually leads to de–coupling of two fields. In order to overcome this problem a special interpolation practiceRhie and Chow [1983],Ferziger and Peric [1997] is required for the mass conserving face velocity in Equation (14.6). The discretized momentum equations can be re–written in a vector form as:

$$\vec{U}_P = \vec{h}_P - \frac{V_P}{a_P} \nabla p_P \, , \; \vec{h}_P = \frac{\sum_j a_j \vec{U}_{P_j} + \vec{S}_{\vec{U}}}{a_P} \tag{14.49}$$

where the source $\vec{S}_{\vec{U}}$ does not include the pressure term. A similar equation for the CV around the cell–face centre ’j’ reads:

$$\vec{U}_j = \vec{h}_j - \overline{\left(\frac{V}{a_P}\right)}_j (\nabla P)_j \tag{14.50}$$

If the variation of the velocity field and the pressure gradient between nodes P and P_j is linear (as would be the case on very fine grids), then vector $\vec{h}_j$, denoted for this case as $\vec{h}_j^{(0)}$, would be given as:

$$\vec{h}_j^{(0)} = \overline{\left(\vec{U}\right)}_j + \overline{\left(\frac{V}{a_P}\right)}_j \overline{(\nabla p)}_j \tag{14.51}$$

By replacing the value of $\vec{h}_j$ in Equation (14.50) by $\vec{h}_j^{(0)}$, the following formula for the face velocity vector is obtained:

$$\vec{U}_j^* = \overline{\vec{U}}_j - \overline{\left(\frac{V}{a_P}\right)}_j \left(\nabla p_j - \overline{\nabla p_j}\right) \tag{14.52}$$

After replacing the pressure gradient ∇p_j by the face gradient definition according to Equation (14.29) , the face velocity vector reads:

$$\vec{U}_j^* = \overline{\vec{U}}_j - \overline{\left(\frac{V}{a_P}\right)}_j \frac{\vec{A}_j}{\vec{A}_j \cdot \vec{d}_j} \left[p_{P_j} - p_P - \overline{\nabla p}_j \cdot \vec{d}_j\right] \tag{14.53}$$

$$\overline{\left(\frac{V}{a_P}\right)}_j = \frac{1}{2}\left(\frac{V_P}{a_P} + \frac{V_{P_j}}{a_{P_j}}\right) \tag{14.54}$$

where the preferred interpolation practice for the face pressure gradient is the arithmetic averaging, $\overline{\nabla p}_j = 0.5(\nabla p_P + \nabla_{P_j})$.

14.2.3 Pressure corrections

The discretized momentum equations are solved for $\vec{U}_P^*$ by using the existing pressure and density fields p^* and ρ^*. The mass fluxes, computed by using the cell–face velocity from Equation (14.53), do not generally satisfy the continuity equation (14.7). A 'mass source' would thus result, defined by:

$$S_m^* = \frac{d}{dt}(\rho_P^* V_P) + \sum_j \dot{m}_j^* \tag{14.55}$$

The basis of the SIMPLE algorithm is to drive this mass source to a negligible small value. This is achieved by introducing the corrections: $\vec{U}_P = \vec{U}_P^* + \vec{U}_P'$, $p_P = p_P^* + p_P'$ and $\rho_P = \rho_P^* + \rho_P'$. The discretized momentum equations for $\vec{U}_P^*$ and $\vec{U}_j^*$ (see Equations (14.49, 14.50)) provide the link between the velocity and pressure corrections:

$$\vec{U}_P' = \vec{h}_P' - \frac{V_P}{a_P}\nabla p_P' \text{ , } \vec{U}_j' = \vec{h}_j' - \overline{\left(\frac{V}{a_P}\right)}_j \nabla p_j' \tag{14.56}$$

With the cell–face gradient $\nabla p_j'$ given by Equations (14.29), the face velocity correction is obtained as:

$$\vec{U}_j' = \vec{h}_j' - \overline{\left(\frac{V_P}{a_P}\right)}_j \frac{\vec{A}_j}{\vec{A}_j \cdot \vec{d}_j}\left(p_{P_j}' - p_P'\right) - \underline{\overline{\left(\frac{V_P}{a_P}\right)}_j \left[\overline{\nabla p'}_j - \frac{\vec{A}_j}{\vec{A}_j \cdot \vec{d}_j}\left(\overline{\nabla p'}_j \cdot \vec{d}_j\right)\right]} \tag{14.57}$$

The mass flux corrections are then given as:

$$\dot{m}_j = (\rho_j^* + \rho_j')(\vec{U}_j^* + \vec{U}_j') \cdot \vec{A}_j = \dot{m}_j^* + \dot{m}_j' \Longrightarrow \dot{m}_j' = \rho_j^* \vec{U}_j' \cdot \vec{A}_j + \rho_j' \vec{U}_j^* \cdot \vec{A}_j + \underline{\rho_j' \vec{U}_j' \cdot \vec{A}_j}$$

Obviously, the last term in the above expression is smaller than others. Next, the density corrections are approximated by:

$$\rho' \approx \left(\frac{\partial \rho}{\partial p}\right)_T p' = C_\rho p' \tag{14.58}$$

where the coefficient C_ρ can be found from the equation of state. For an ideal gas we have: $C_\rho = 1/(R_g T)$ The cell–face density correction ρ'_j is approximated here by a fraction of the upwind scheme:

$$\rho'_j = \left[C_{\rho P}\max\left(\dot{m}^*_j, 0\right)p'_P - C_{\rho P_j}\max\left(-\dot{m}^*_j, 0\right)p'_{P_j}\right]\frac{\beta_p}{\dot{m}^*_j} \tag{14.59}$$

where we use $\beta_p = \alpha_p$, α_p being the under–relaxation factor for pressure. It should be noticed that such a choice has no influence on the final solution since the pressure corrections tend to be close to zero at convergence.

We recognise that the velocity corrections from neighbouring cells, represented by $\vec{h}'_P$ ($\vec{h}'_j$) are not known at this stage. In the SIMPLE algorithm they are neglected, which is a crude oversimplification. It is also common practice to neglect the underlined term in Equation (14.58) and the contribution to the mass flux correction from the underlined term in Equation (14.57). The latter is associated with non–orthogonal grids. With these approximations and mass flux definitions $\dot{m}_j = \dot{m}^*_j + \dot{m}'_j$, the mass balance equation (14.7) is reduced to:

$$\frac{d}{dt}\left(C_{\rho P}\, p'_P V_P\right) + \sum_{j=1}^{n_f} \dot{m}'_j = -S^*_m \tag{14.60}$$

>From the above equation, the pressure correction equation is derived:

$$a'_P p'_P = \sum_{j=1}^{n_f} a'_{P_j} p'_{P_j} - S^*_m \tag{14.61}$$

which has a similar form as a general discretized equation (14.48). The distinctive features of the pressure correction equation are discussed inDemirdzic et al. [1993],Ferziger and Peric [1997]. After solving the algebraic equations for p' (before that p' is initialised zero everywhere), the cell velocities, pressure and mass fluxes are corrected as discussed above. The pressure is corrected only by a fraction of p': $p_P = p^*_P + \alpha_p p'_P$, where α_p is typically 0.1–0.2.

The effect of the grid non-orthogonality (the last term in Equation (14.57)) can be now taken into account by performing the second correction stepFerziger and Peric [1997] with $\vec{U}_P = \vec{U}^*_P + \vec{U}'_P + \vec{U}''_P$, $p_P = p^*_p + p'_P + p''_P$, $\dot{m}_j = \dot{m}^*_j + \dot{m}'_j + \dot{m}''_j$. The new corrections for the face velocity and mass flux are obtained from Equations (14.57) and (14.58) as follows:

$$\vec{U}''_j = -\overline{\left(\frac{V_P}{a_P}\right)}_j \frac{\vec{A}_j}{\vec{A}_j\cdot\vec{d}_j}\left(p''_{P_j} - p''_P\right) - \overline{\left(\frac{V_P}{a_P}\right)}_j \left[\overline{\nabla p'}_j - \frac{\vec{A}_j}{\vec{A}_j\cdot\vec{d}_j}\left(\overline{\nabla p'}_j\cdot\vec{d}_j\right)\right] \tag{14.62}$$

$$\dot{m}''_j = \rho^*_j \vec{U}''_j\cdot\vec{A}_j + \rho''_j \vec{U}^*_j\cdot\vec{A}_j \tag{14.63}$$

where the second density correction ρ''_j is given by Equation (14.59) with p' replaced by p''. The mass balance, Equation (14.7), is already enforced by the corrected mass fluxes $\dot{m}^*_j + \dot{m}'_j$. Therefore, the second pressure correction equation can be derived from the following balance equation:

$$\frac{d}{dt}\left(C_{\rho P}\, p''_P V_P\right) + \sum_{j=1}^{n_f} \dot{m}''_j = 0 \tag{14.64}$$

Notably, the second pressure correction equation has the same coefficient matrix as the equation for p'. Its inclusion can be beneficial in terms of improved convergence for highly non–orthogonal grids, i.e. for the grids with angles between $\vec{A}_j$ and $\vec{d}_j$ less than 45°.

14.2.4 Boundary conditions for pressure corrections

At all boundaries with known velocities the mass flux corrections are zero: $\dot{m}_b = \dot{m}_b^*$, $\dot{m}_b' = 0$. Therefore, at these boundaries the boundary coefficients a_b' are zero. In other cases, the boundary velocities and mass fluxes need to be corrected. In fully compressible flows ($Ma > 0.3$) this is the case at open flow boundaries (inlets, outlets, free-streams) where the pressure is either directly or indirectly prescribed. At these boundaries, the flow is subsonic for the Mach number at boundary $Ma_b < 1$ and supersonic for $Ma_b > 1$ where the Mach number is defined as

$$Ma = \frac{|\vec{U}|}{c} \ , \ c = \sqrt{\gamma R_g T} \tag{14.65}$$

with c being the speed of sound and γ is the isentropic exponent ($\gamma = 1.4$ for an ideal gas). Here we outline the implementation of boundary conditions at the inlets and outlets. In the spirit of SIMPLE, we need to express the boundary mass flux and its correction as a function of the pressure and pressure correction, respectively. The mass flux corrections lead to the boundary pressure correction coefficient a_b' and upon solution of the pressure correction equation, the boundary pressure and mass flux are corrected as: $p_b = p_b^* + \alpha_p p_b'$, $\dot{m}_b = \dot{m}_b^* + a_b' p_b'$

Subsonic Inlet. The total or stagnation boundary condition is valid for inlets. Thus, the total pressure p_t, total temperature T_t and the flow direction $\vec{i}_b = \vec{U}_b / |\vec{U}_b|$ are defined and the pressure is extrapolated from the inside. From the following isentropic relations:

$$p_t = p_b \left(1 + \frac{\gamma - 1}{2} Ma_b^2\right)^{\frac{\gamma}{\gamma-1}} \tag{14.66}$$

$$T_t = T_b \left(1 + \frac{\gamma - 1}{2} Ma_b^2\right) \tag{14.67}$$

one calculates the boundary Mach number Ma_b and temperature T_b, respectively. Then, the magnitude of the velocity vector can be obtained from Equation (14.65) as :

$$|\vec{U}_b|^2 = \frac{2\gamma R_g T_b}{\gamma - 1} \left[(p_t/p_b)^{\frac{\gamma-1}{\gamma}} - 1 \right] \tag{14.68}$$

Differentiating the above equation with respect to pressureDemirdzic et al. [1993], the mass flux correction at the boundary $\dot{m}_b' \approx \rho_b^* (\vec{i}_b \cdot \vec{A}_b)(\partial |\vec{U}| / \partial p)_b$ can be derived as follows:

$$\dot{m}_b' \approx -\dot{m}_b^* \frac{R_g T_b^*}{|\vec{U}_b^*|^2 p_t} \left(1 + \frac{|\vec{U}_b^*|^2}{C_{pb} T_b^*}\right)^{\frac{2\gamma-1}{\gamma-1}} p_b' \tag{14.69}$$

Assuming $p_b' = p_P'$, the pressure correction coefficient a_b' can be easily defined from the above equation.

Supersonic Inlet. If the flow is supersonic at the inlet all flow variables: p_t, T_t, p_b, $\vec{i}_b$ (the density is calculated from the equation of state) must be specified.

Subsonic Outlet. When the outlet is subsonic, the pressure is fixed and all other variables are extrapolated. The boundary velocity is obtained from Equation (14.53) applied to a boundary face. This velocity and the corresponding velocity correction are given as:

$$\vec{U}_b^* = \vec{U}_P^* - \frac{V_P}{a_P} \frac{\vec{A}_b}{\vec{A}_b \cdot \vec{d}_b} \left(p_b - p_P^* - \nabla p_P^* \cdot \vec{d}_b \right) \tag{14.70}$$

$$\vec{U}_b' \approx - \frac{V_P}{a_P} \frac{\vec{A}_b}{\vec{A}_b \cdot \vec{d}_b} \left(p_b' - p_P' \right) \tag{14.71}$$

where the boundary pressure correction is zero, $p_b' = 0$. The mass flux correction at the pressure boundary then reads (see Equations (14.58, 14.59)):

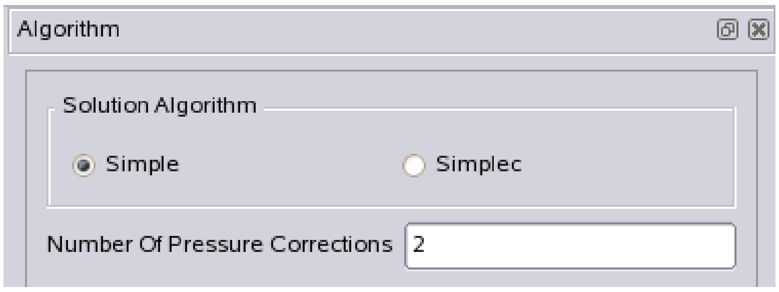
$$\dot{m}_b' \approx \rho_b^* \vec{U}_b' \cdot \vec{A}_b + \frac{\alpha_P}{\rho_b^*} C_{\rho P} \max\left(\dot{m}_b^*, 0\right) p_P' \tag{14.72}$$

and serves to derive the pressure correction coefficient a_b'. The above treatment of pressure boundaries is applicable to both compressible and incompressible flows.

Supersonic Outlet. At supersonic outlets, all variables including the pressure are extrapolated from the upstream. However, the computations usually start with the initial fixed pressure until establishing the stable supersonic conditions in the outlet region.

14.2.5 Setting up pressure–correction algorithm

For each Fluid Domain node in the Solver Setup Tree panel, left-click on Algorithm node and the Algorithm panel is displayed to the right as in Figure 14.9. Now the Solution Algorithm can be selected to either be Simple or Simplec . Number of Pressure Corrections can also be specified for each solution algorithm. The default value is 2.


Algorithm

Solution Algorithm

[✓ Simple] Simplec

Number Of Pressure Corrections: 2

Figure 14.9: *R–Desk* setup for pressure–correction algorithm

14.2.6 Linear solvers

The bandwidth of the sparse matrix is reduced by applying the Reverse Cuthill–McKee re–ordering algorithm to cells, George and Liu [1981]. The system of linearised algebraic equations (14.48) is under–relaxed implicitly, Patankar [1980]. It is solved by a family of preconditioned conjugate gradient solvers, Van Der Vorst [1992]; Sleijpen and Fokkema [1993].

Setting up linear solver. >From *R–Desk* under each Fluid Domain node in the Solver Setup Tree panel, left-click on Equations & Solver node and the Equations_Solver panel is displayed to the right.

The Solver Type can be specified for each transport equation to either be the Bi-cg Stabilised method or Symmetric Conjugate Gradient method (Figure 14.5). The default solver is Bi-cg Stabilised .

A number of pre-conditioners are also available such as zero and first order incomplete lower-upper (ILU) factorisation pre-conditioners, Meijerink and Van Der Vorst [1981]; Benzi [2002], Jacobi, including a new unstructured grid adaptation of the well–known Stone's SIP solver as an efficient pre-conditioner. The list of Preconditioning options are given in Figure 14.5.

A number of ILU Preconditioning Options for Parallel Runs are also provided such as: Localised , Mixed and Global . The Number of Inner Solver Iterations and Number of Preconditioning Iterations can be specified directly including the Solver Tolerance value.

14.3 Implementation of boundary conditions

Considering boundary faces, the convective and diffusion fluxes are calculated in the same manner as for the inner cell–faces. The convective fluxes are evaluated by using the upwind scheme. In case of diffusion flux, Equation (14.33) is modified as:

$$D_b = \Gamma_{\phi b} \frac{A_b^2}{\vec{A}_b \cdot \vec{d}_b} \left(\phi_b - \phi_P\right) + \underline{\Gamma_{\phi b} \nabla \phi_P \cdot \left(\vec{A}_b - \frac{A_b^2}{\vec{A}_b \cdot \vec{d}_b} \vec{d}_b \right)} \tag{14.73}$$

where the subscript "b" signifies the boundary face. The above equation accommodates both Dirichlet conditions (specified boundary values) and Neumann conditions (prescribed boundary flux). In the latter case it is used to compute the boundary values. In case of momentum and energy, the diffusion coefficients at wall boundaries $\Gamma_{\phi w}$ are defined by Equations (9.40).

14.4 Poor Quality Cell Treatment

In VECTIS-MAX , poor quality cells are identified applying the following criteria:

- □ Maximum cell–face and boundary face skewness angle, see Equation (14.34)
- □ Minimum volume change ratio: this is defined as the ratio between a cell volume and the maximum volume among its neighbours.
- □ Minimum wall distance ratio for near–wall cells: this is an equivalent to the cell aspect ratio. Here, it represents the ratio of the wall normal distance (distance from the cell centre to the centre of the representative wall) and an average normal distance of cell neighbours to the same wall.

Typical values for the maximum face angle, minimum volume change and minimum wall distance ratio are 75°, 0.01 and 0.15, respectively.

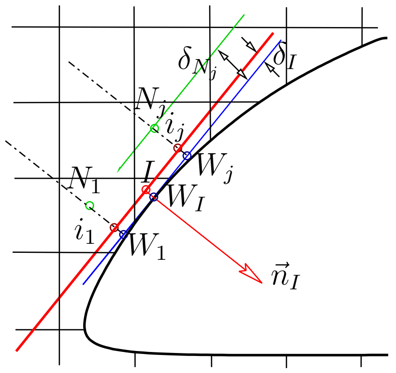


Figure 14.10: Schematic representation of interpolative scheme

The Cell Interpolative Scheme (CIS) presented here excludes poor quality cells from the solution of discretized equations. Instead, flow variables at these cells are determined from an interpolation of neighbouring cell values and imposed boundary condition values. A sketch in Figure 14.10 shows the poor quality cell I, adjacent to the representative wall W_I, with two ($j = 2$) neighbours N_j. For the purpose of interpolation, neighbours are defined as the surrounding cells that are further away from the wall than cell I. The 'first' neighbours (those that have common cell–faces with the cell I) are considered. The poor quality neighbouring cell will be a qualifying neighbour only if the cell I does not have any other neighbours.

There are two stages in the interpolation procedure which have some similarity with the interpolation algorithm of Kalitzin and Iaccarino [2003]. First, neighbouring nodes are projected normally onto an interpolation plane, parallel to the wall and passing through the cell I. The state of variables at new, intermediate plane locations i_j is constructed directly from the states of qualifying neighbouring cells N_j and wall nodes W_j, $j = 1, n_I$, n_I being a number of qualifying neighbours. In the next step, an interpolation based on the inverse–distance weighting factors is designed to calculate variables at the cell I:

$$\phi_I = \sum_{j=1}^{n_I} (\alpha_i)_j (\phi_i)_j \text{ with } (\alpha_i)_j = \frac{1}{\overline{i_j,I}} / \sum_{j=1}^{n_I} \left(\frac{1}{\overline{i_j,I}} \right) \tag{14.74}$$

where $\overline{i_j,I}$ represents the distance between intermediate nodes i_j and the cell I.

The type of interpolation used to obtain values at the interpolation plane locations depends on the near-wall behaviour of the considered variable. First, we assume that flow variables at wall nodes W_j have the same values as variables at the wall node W_I. Considering the intermediate node i (subscript j is omitted), the velocity vector is split into components parallel and normal to the wall,$\vec{U_{i,p}}$ and $\vec{U_{i,n}}$, respectively:

$$\vec{U}_i = \vec{U}_{i,p} + \vec{U}_{i,n} \text{ with } \vec{U}_{i,n} = \left(\vec{U}_i \cdot \vec{n}_I \right) \vec{n}_I \text{ , } \vec{U}_{i,p} = \vec{U}_i - \vec{U}_{i,n} \tag{14.75}$$

Similar decomposition is used for adjacent nodes N and W. The parallel velocity can be calculated from the universal near–wall profile, Equation (**??**), assuming the constant wall shear stress τ_w. Employing the effective wall viscosity μ_w, Equation (9.40), we can write:

$$\tau_w = \mu_i \frac{\Delta U_{i,p} Y_i^*}{U_i^* \delta_I} = \mu_N \frac{\Delta U_{N,p} Y_N^*}{U_N^* \delta_N} \tag{14.76}$$

where $\Delta U_{i,p}$ and $\Delta U_{N,p}$ are given as $\Delta U_{i,p} = U_{W,p} - U_{i,p}$ and $\Delta U_{N,p} = U_{W,p} - U_{N,p}$, respectively. The above equation defines the wall parallel velocity ratio $\varphi_{i,U}$ as:

$$\varphi_{i,U} = \frac{\Delta U_{i,p}}{\Delta U_{N,p}} = \frac{\delta_I}{\delta_N} \frac{\mu_N}{\mu_i} \frac{Y_N^* U_i^*}{Y_i^* U_N^*} \tag{14.77}$$

The direction of the parallel velocity at i is taken to be that of the parallel velocity at the neighbour node N. For the wall normal velocity, the parabolic profile is physically sound:

$$\Delta U_{i,n} = \Delta U_{N,n} \left(\frac{\delta_I}{\delta_N} \right)^2 \text{ with } \Delta U_{i,n} = \left(\vec{U}_W - \vec{U}_i \right) \cdot \vec{n}_I \text{ , } \Delta U_{N,n} = \left(\vec{U}_W - \vec{U}_N \right) \cdot \vec{n}_I \tag{14.78}$$

The resulting velocity vector at intermediate nodes i can be expressed as:

$$\vec{U}_i = \varphi_{i,U} \vec{U}_N + (1 - \varphi_{i,U}) \vec{U}_W + \left\{ \varphi_{i,U} - (\delta_I / \delta_N)^2 \right\} \vec{n}_I \tag{14.79}$$

A similar procedure, employing the wall effective conductivity λ_w, Equation (9.40), and assuming the constant heat flux q_w across the near–wall region, leads to an expression for the temperature:

$$T_i = \varphi_{i,T} T_N + (1 - \varphi_{i,T}) T_W \tag{14.80}$$

where $\varphi_{i,T}$ denotes the temperature ratio

$$\varphi_{i,T} = \frac{T_W - T_i}{T_W - T_N} = \frac{\delta_I}{\delta_N} \frac{\lambda_N Pr_N}{\lambda_i Pr_i} \frac{Y_N^* T_i^*}{Y_i^* T_N^*} \tag{14.81}$$

and the dimensionless temperature distribution T^* should be known in terms of the Y^*, see Equation (9.43).

The main difficulty here is how to determine the turbulent kinetic energy k at intermediate nodes in order to evaluate the non-dimensional wall distances Y^*, in particular if the value of k at these nodes is around the maximum. A crude approach is to use wall functions for the k as designed in Kalitzin et al. [2005] and Absi and Bennacer [2006]. According to the latter reference, k is a function of the wall distance $Y^+ = \rho U_\tau Y/\mu$ and we can write:

$$k_i = B\frac{(\tau_w)_N}{\rho_N}(Y_i^+)^2 e^{-Y_i^+/A^+} \tag{14.82}$$

where $B \approx 0.14$ and $A^+ \approx 8$ are the constants. With known k at the poor quality cell I, the dissipation rate can be specified without an interpolation, see Equation (9.47)

Note that in case of laminar flow or viscous sub-layer, the above interpolative functions for the state of variables at intermediate nodes i are reduced to the linear ones. For the poor quality cells not adjacent to the walls, there is no interpolation plane and Equation (14.74) is used in conjunction with the qualifying neighbour and possible boundary values. The pressure and mass conservation equation do not require any special treatment at poor quality cells. The interpolative scheme is applied after solving the linearised discretized equations and after that interpolated variables are used in the same way as variables at normal cells.

Switching On/Off poor quality cell treatment option: From the Solver Setup Tree under Fluid Domain / Solid Domain , left-click on Discretise . This opens up the Discretise panel as seen in Figure 14.7. Under Bad Cell Treatment a number of options are available from the ListBox (see Figure 14.7). The Off option simply switches off the cell treatment option. To apply cell treatment option to all varaibles except pressure correction then the user should select All Variables Except Pressure Correction . The user can also choose to apply bad cell treatment to pressure correction as well but not the 1st pressure correction by selecting All Variables Except 1st Pressure Correction .

14.5 Parallelisation

The whole solution process is parallelised, including above pre-conditioners, by employing domain decomposition strategy and MPI–based message passing.

15 MODELLING RADIATION

15.1 Introduction And Overview

Thermal radiation between different surfaces in a flow domain model can be predicted using the VECTIS radiation module. VECTIS calculations can therefore include the heat transfer by convection and conduction and radiation. This help manual describes the radiation module and its individual programs in terms of what they do and how to use them. This document assumes that the user has a basic knowledge of using of VECTIS.

The VECTIS radiation module is currently based on diffuse radiation theory so that the emission and reflection of radiation is independent of direction. The distribution of the radiated heat to the surrounding surfaces is determined by the view factors of each patch. Only wall-to-wall radiation is currently modelled and the fluid between the surfaces is assumed to have no interaction.

15.2 Mesh File Setup

The radiation modules require that the input mesh is gap-free (between faces). To ensure this, it advisable to preprocess the mesh file with *vpre* in order to eliminate any topological gaps within the boundaries. For the case of multi-domain meshes, this should be repeated for each single domain grid prior to coalescing stage. See below for examples:

Single domain:

```
vpre -close-boundary-gaps mesh1.GRD
```

Multi-domain, 3 domains:

```
vpre -close-boundary-gaps mesh1.GRD
vpre -close-boundary-gaps mesh2.GRD
vpre -close-boundary-gaps mesh3.GRD
vpre -jtype 0 mesh1.GRD mesh2.GRD mesh3.GRD -o multidom.GRD
```

15.3 Radiation Setup

In order to run VECTIS-MAX with radiation modelling on, it is first necessary to set-up various files required by the radiation modules. The first stage involves preparing a *radfile* required by the two modules *radprep* and *radvfm* , and later the radiation solver within *vsolve* . This *radfile* is a simple text file which is automatically created upon saving the main solver input file (if radiation is activated).

The radiation panel can be navigated to from within the "Global Domain" panel (Fig 15.1). This panel (Fig 15.2) is used to set-up global quantities pertaining to a radiation problem.

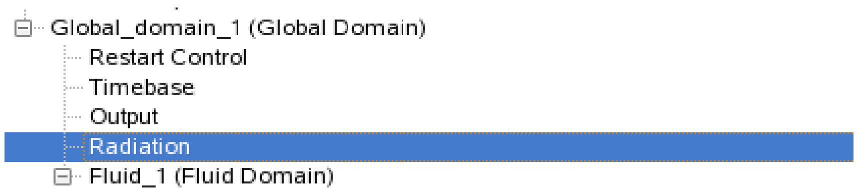


Figure 15.1: Global radiation location within the solver-setup tree.

The *Radiation Modelling* pull-down menu currently has only two options, "none" and "Surface to surface". The *Radiation File* groupBox is used to specify the name of the input file (*radfile*) to be subsequently used by the radiation modules. The "Start" box indicates the time step/iteration number when the radiation solving should begin. The "Frequency" box denotes how often (time step/iteration) the radiation solver should be called.

Global Surface-to-surface Parameters

PAT, VFM & CON filenames

These are auxillary filenames which will normally adopt the base name of the mesh file in the current input file. The user can however freely override this. The PAT file is generated when *radprep* is run, and contains superpatch information. The CON is the corresponding connectivity file. The VFM file is created by *radvfm* and contains the superpatch view-factor matrix.

Reflectivity

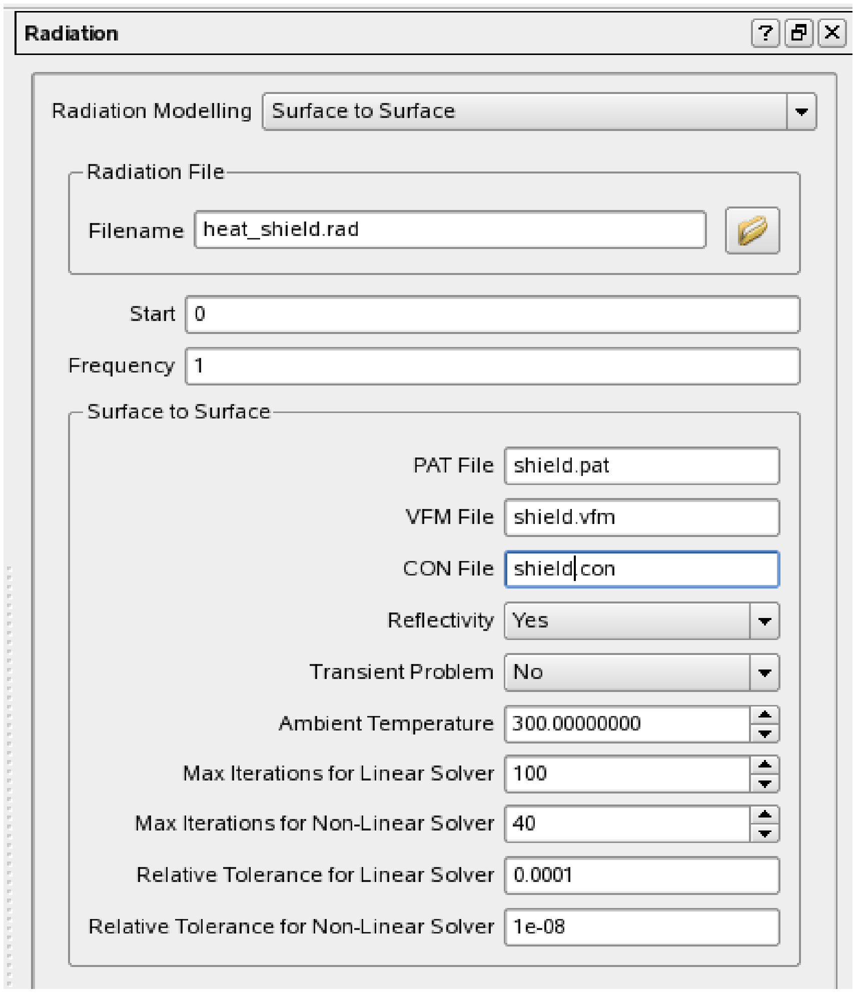
This reflectivity option allows for the inclusion of secondary radiation i.e. radiation that is emitted from a surface that is also receiving radiation heat flux. By default this is set to 'YES' so that radiation can be reflected from a surface.

Transient Problem

The transient option is used to define if the radiation analysis will be for a transient or steady state analysis.

Default Ambient Temperature

This value is going to be used whenever a super patch is not fully surrounded by patches i.e. its view factor summation is less than 1. In such case, the radiation from the background needs to be included. The ambient temperature is needed because in such a case where the super patches view factor summation is less than 1, the patch is not only losing energy by radiating to the ambient, but


Radiation

Radiation Modelling: Surface to Surface

Radiation File
Filename: heat_shield.rad

Start: 0
Frequency: 1

Surface to Surface
PAT File: shield.pat
VFM File: shield.vfm
CON File: shield.con
Reflectivity: Yes
Transient Problem: No
Ambient Temperature: 300.00000000
Max Iterations for Linear Solver: 100
Max Iterations for Non-Linear Solver: 40
Relative Tolerance for Linear Solver: 0.0001
Relative Tolerance for Non-Linear Solver: 1e-08

Figure 15.2: *R–Desk* radiation setup panel.

it also receives some energy back from the background.

This default ambient temperature is used for all radiating boundaries unless a boundary has its own ambient temperature specified in the Radiation Boundary panel.

Solver control parameters The 4 remaining parameters control the radiation solver (radsolv) and are fairly self-explanatory. The default values chosen should normally be fine.

Boundary Specific Surface-to-surface Parameters

The selection of the participating boundaries is done from within the boundary panels as shown in Figs 15.3, 15.4.

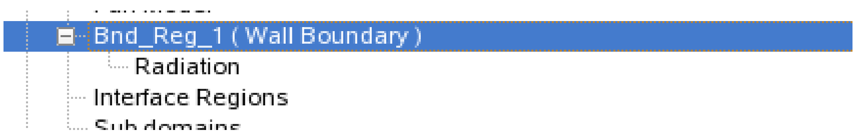
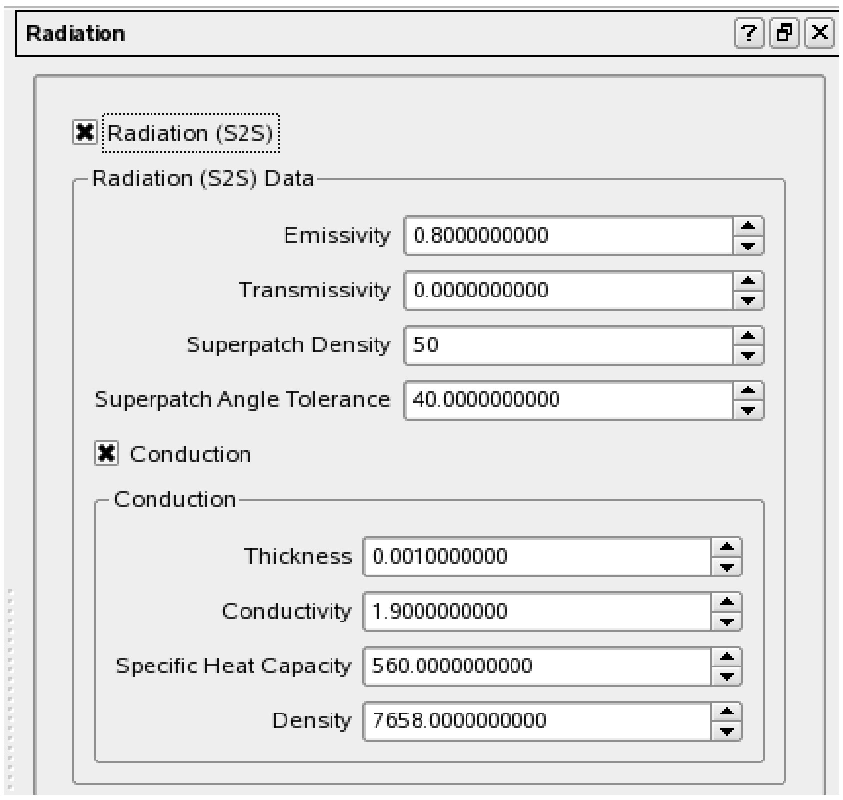

Bnd_Reg_1 (Wall Boundary)
Radiation
Interface Regions
Sub domains

Figure 15.3: Radiation boundary panel location within solver-setup tree.

The radiation panel only appears for wall boundaries.


Radiation

[✓] Radiation (S2S)

Radiation (S2S) Data

Emissivity	0.8000000000
Transmissivity	0.0000000000
Superpatch Density	50
Superpatch Angle Tolerance	40.0000000000

[✓] Conduction

Conduction

Thickness	0.0010000000
Conductivity	1.9000000000
Specific Heat Capacity	560.0000000000
Density	7658.0000000000

Figure 15.4: *R–Desk* radiation boundary panel.

The "Radiation (S2S)" check-box activates a particular radiation boundary. If a boundary is activated, then *radprep* , *radvfm* and the radiation solver will perform the relevant calculations for the patches and super patches belonging to this boundary set. If it is inactive and *radprep* and *radvfm* are run then the boundary will not be included in the super patch construction and view factor calculation respectively and hence this boundary can not be included in the radiation calculation without re-running *radprep* and *radvfm* with it activated. If a boundary is set as active and *radprep* and *radvfm* are run so that the boundaries super patches and view factors are calculated, the boundary can then be set as inactive before running *vsolve* so that this boundary is then not included in the radsolv calculation.

Emissivity

This controls the emissivity of the boundary (default: 0.8). This value ranges from $0 \rightarrow 1$.

Transmissivity

This controls the transmissivity of the boundary (default: 0.0). This value ranges from $0 \rightarrow 1$, where 0.0 indicates complete opaqueness, and 1.0 total transparency. NOTE: if the boundary is material interface, then any other interfaces attached to this transparent material will automatically be updated.

Superpatch density

This is the number of superpatches per boundary (default: 50), if -1 one superpatch per patch. These superpatches are typically a coarser representation of the boundary. The lower the value is, the quicker (but coarser) the solution will be.

Superpatch Angle Tolerance

This is the maximum allowable angle between patches within a superpatch (default: 40°). With this option *radprep* will construct the super patches for the boundary and try to ensure that the total number of super patches is as near to the defined limit as possible whilst also using the angle-limit value to define the maximum angle between the normal vectors of two neighbouring patches. i.e if two neighbouring patch normal angles differ by more than the angle-tolerance value then they will be grouped into different super patches.

Conduction

If this check-box is activated, then the surface conduction heat flux for this boundaries super patches will be solved. The following thermal property data should be defined:

Thickness

This is the thickness of the super patches $[m]$. Default: $10^{-3}m$.

Density

This is the material density $[kgm^{-3}]$.

Heat Capacity

Material heat capacity $[Jkg^{-1}K^{-1}]$.

Conductivity

Material heat conductivity $[Wm^{-1}K^{-1}]$.

Once the *radfile* has been generated (this is automatically done after saving the input file), it is then necessary to generate other auxillary files required by the radiation solver module. This is currently done manually (via the command line). See below.

15.3.1 Running of *radprep*

The *radprep* program is used to generate the *patfile* which contains the superpatch connectivity. The radprep module can be run in two ways:

```
    radprep [options see below] radfile
```

Available options:

-nosc	Connectivity for surface conduction will not be computed.
-notc	Connectivity for through conduction will not be computed.
-sn	Superpatches's normals will be switched to opposite direction for finding connectivity for through conduction.
-minangle Angle	Minimum view angle for finding through conduction connectivity. Default: minangle = 80.0 degrees.
-maxangle Angle	Maximum view angle for finding through conduction connectivity. Default: maxangle = 89.0 degrees.
-ntcs Number	Number of connection to store for through conduction connectivity search. Default: ntcs = 3.
-v	Shows version of this program.
-h	Shows this help.

When radprep is run in this way the program does the following:

1. Reads the radfile and finds following information:
   - □ name of mesh file.
   - □ name of patfile which should be created.
   - □ universal boundary numbers for the boundaries which should be taken into account for radiation and requirements for the super patch generation for each of these boundaries.
2. Reads the VECTIS phase4 mesh file.
3. Reads the phase4 mesh surface patch connectivity file (*CON confile). If the confile does not exist radprep determines the connectivity of the phase4 mesh surface patches and generates the confile.
4. Assembles the required number of super patches for each boundary as defined in the radfile.
5. Assembles the connectivity information for the created super patches
6. Determines the surface conduction connectivity
7. Determines the normal conduction connectivity
8. Writes the patfile

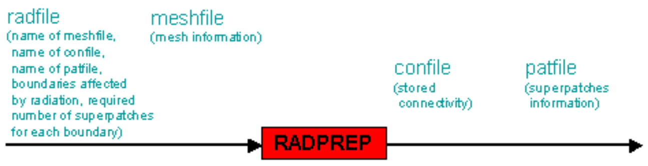

radfile
(name of meshfile, name of confile, name of patfile, boundaries affected by radiation, required number of superpatches for each boundary)
meshfile
(mesh information)
RADPREP
confile
(stored connectivity)
patfile
(superpatches information)

Figure 15.5: Scheme of the radprep program.

See below for the description of minimum angle and maximum angle.

NOTE that the universal boundary regions that are setup in the VECTIS phase1 pre-processor should be identified so that one boundaries triangles are connected and not separate from each other. So there should not be separate clusters of triangles that are identified as the same boundary number that are not connected and hence all triangles belonging to one boundary number should be connected.

See below for the description of minimum angle and maximum angle.

Minimum Angle and Maximum Search Angle and number of connection settings

The "-minangle" angle and "-maxangle" angle switches are used to set the minimum and maximum search angles for the normal conduction connectivity calculation.

The default values for the minimum search angle that will be used if it is not set using the command option is 80 degrees and the maximum search angle is the minimum angle + 5 degrees with a maximum limit of 89 degrees. The definition of the search angles are shown by the figure below.

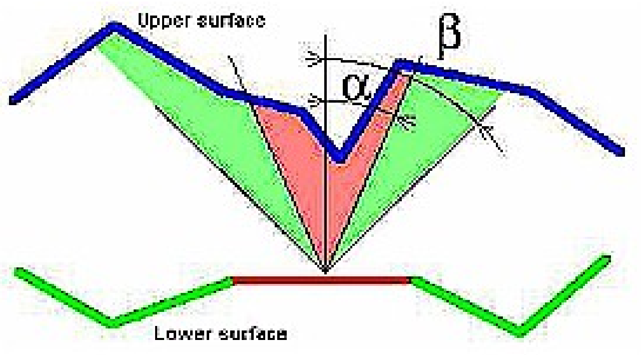


Figure 15.6: Figure showing the definition of the normal conduction minimum (α) and maximum (β) search angles.

The "-ntcs" Number switch that can be used at the command line allows for the definition of the maximum number of super patches that an individual super patch can connect to for the normal conduction. By default the number of connections is set to 3. Therefore if the number of connections is set to 3 the best three normal connections for each super patch will be used. Refer to the Radprep theory section for information about how the best connections are determined.

15.3.2 Running of *radvfm*

After the *radprep* program has completed the view factor matrix should then be created using the following command:

```
radvfm [-r] [-s*] [-v] [-h] radfile
```

where

-v	Shows version of the program. If this parameter occurs among arguments the other parameters are omitted.
-s*	Allows to set the number of used pixels in hemispere. The asterisk * stands for 1-5 numbers. For example, if -s3 is used, the number of pixels on the hemisphere diameter will be 300 (3x100). This means that the hemisphere will be placed on the square 300x300 pixels. The level 3 is default.
-r	Reverses the boundary orientation (used only when a trifile is used as the input geometry).
-h	A help similar to the information above is shown.

The radvfm program does the following:

1. Reads the radfile and determines the following information:
   - □ name of mesh file (or tri file)
   - □ name of patfile
   - □ name of vfmfile which should be created
2. Reads the phase4 mesh file (or tri file)
3. Reads the super patch file (patfile)
4. Allocates the necessary memory space
5. Loops over all the super patches and calculates the rows of the view factors matrix. These rows are directly written to vfmfile using the Ricardo SDF file format.

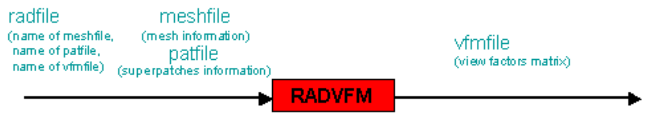


Figure 15.7: Scheme of the radvfm program.

15.4 Radvfm Theory

15.4.1 Assembling the view factors matrix

To assemble the view factors matrix the direct visibility of each patch to every other patch must be examined. If the two super patches can directly see each other, the view factors Fij and Fji can be calculated for them.

15.4.2 Calculation of view factor

The calculation of view factor between two patches can be done using this simplified formula:

$$F_{ij} = \frac{\cos\beta_i \cos\beta_j}{\pi r^2} A_j \tag{15.1}$$

where

F_{ij}	view factor [-] between super patches i and j
β_i and β_j	angles [rad] between the position-dependent normal vectors to super patches i and j
r	length of a line connecting the points of evaluation of the normal
A_j	area $[m^2]$ of the super patch j

The situation is visualized below.

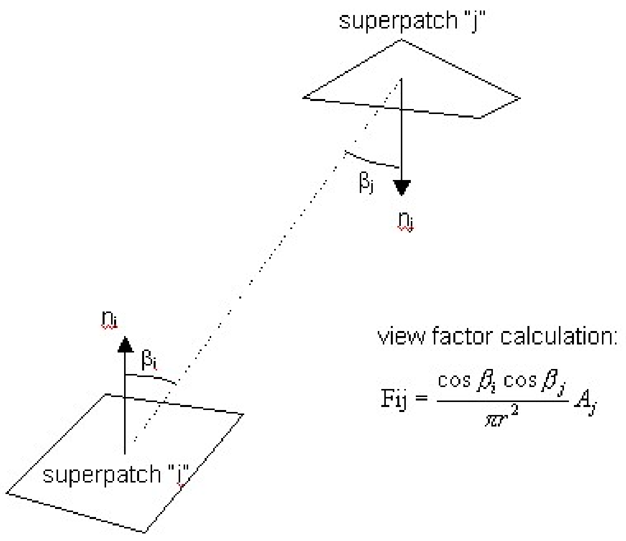


Figure 15.8: Heating super patch (i) and heated super patch (j) in view factor calculation.

15.4.3 Calculation of Super patch view factors

15.4.3.1 Hemisphere-Base Projection Methods

In order to calculate all view factors for an input geometry, the Nusselt analog method is used in radvfm. For each super patch, all patches in the visible half space are projected onto the hemisphere. The radius of the hemisphere is unity. Firstly, each patch is spherically projected to the hemisphere. Secondly, the projection is further cylindrically projected to the base of the hemisphere. The form factor is calculated as the area projected on the base of the hemisphere divided by the area of the whole base of the hemisphere. The pictures below show the described principal of the Nusselt analog. The first picture shows a 2D situation; the second picture illustrates a situation in 3D.

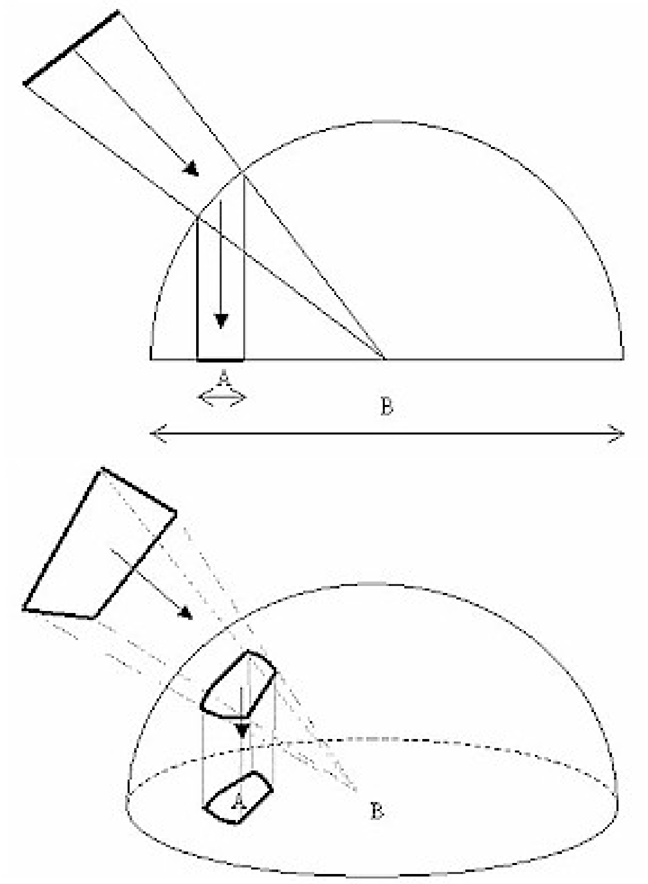


Figure 15.9: Figures showing the Hemisphere projection used in the radvfm calculation.

The hemisphere is based on a square formed by pixels. The pixels outside the hemisphere are not used. The relevant pixels are coloured by projected patches when objects which are nearer eclipse the objects which are far. When all patches are projected, the number of pixels painted by the same colour are counted. View factor of processed super patch i and an other super patch j can be counted as:

$$VF_{ij} = \frac{4N_j}{\pi D^2} \tag{15.2}$$

where

N_j is number of pixels painted by the colour of the super patch j.
D is number of pixels on the hemisphere diameter (the length in pixels of the side of the square onto which is the hemisphere placed).

15.5 Radsolv Theory

Heat Transfer Theory

The governing equation for the heat transfer is shown below by Eq (15.3) .

$$\rho c \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) \tag{15.3}$$

Finite Volume Method

The finite volume method is based on flux balance. It means, that sum of fluxes which enters the system must be equal the sum of fluxes which leaves the system.

Using Green's theorem, equation 15.3 can be transformed to:

$$\int_\Omega \rho c \frac{\partial T}{\partial t} d\Omega = \int_\Gamma k \frac{\partial T}{\partial n} d\Gamma \tag{15.4}$$

In this transformation the volume integral was replaced by the integration over the boundary.

There are several types of boundary conditions that the heat-transfer equation should allow so that the different types of heat-transfer problems can be modelled. The above equation only allows for a heat flux boundary condition. Another boundary condition type is a fixed temperature boundary condition. This is done by simply replacing the unknown function T in equation 15.4 with its known value and adding it to the right hand side as a source. This approach is described in more details in the Discretization section.

Boundary Conditions

$$-k \frac{\partial T}{\partial n} = k \frac{T - T_n}{r} = q_k \quad \text{on } \Gamma_1 \tag{15.5}$$
$$-k \frac{\partial T}{\partial n} = h \frac{T - T_\infty}{r} = q_c \quad \text{on } \Gamma_2 \tag{15.6}$$
$$-k \frac{\partial T}{\partial n} = \varepsilon (T^4 - T_\infty^4) = q_r \quad \text{on } \Gamma_3 \tag{15.7}$$

where

Γ_1	boundary on which heat flux is specified
Γ_2	boundary on which convective heat loss is specified
Γ_3	boundary on which radiation is specified

Discretization

$$\int_\Gamma \rho c \frac{\partial T}{\partial t} d\Omega = \int_{\Gamma_1} q_k d\Gamma + \int_{\Gamma_2} q_c d\Gamma + \int_{\Gamma_3} q_r d\Gamma \tag{15.8}$$

Equation 15.8 can not be solved directly, therefore it needs to be discretized into discrete points. Because the Finite Volume Method is used in radsolv the discrete points are the center points of the patches or super patches.

The equation system then looks like:

$$M\dot{T} + KT + CT + RT^4 = Q + C_\infty + R_\infty \tag{15.9}$$

where

$$M\dot{T} = \int_\Omega \rho c \frac{\partial T}{\partial t} d\Omega = \delta_{ij}\rho_i c_i V_i \dot{T}_i \quad - \quad \text{mass matrix} \tag{15.10}$$

$$KT = \int_\Gamma k \frac{T - T_n}{r} d\Gamma = \sum_{j=1}^{nb_1} k \frac{A_{ij}}{r_{ij}} (T_i - T_j) \quad - \quad \text{conduction matrix} \tag{15.11}$$

$$CT = \int_\Gamma h \cdot T d\Gamma = \sum_{j=1}^{nb_2} h_{ij} A_{ij} T_j \quad - \quad \text{convection matrix} \tag{15.12}$$

$$RT^4 = \int_\Gamma \varepsilon \sigma T^4 d\Gamma = \sum_{j=1}^{nb_3} \varepsilon_{ij} \sigma A_{ij} T_j^4 \quad - \quad \text{radiation matrix} \tag{15.13}$$

$$Q = \int_\Gamma q d\Gamma = \sum_{j=1}^{nb_1} q_{kij} A_{ij} \quad - \quad \text{conduction (heat flux) matrix} \tag{15.14}$$

$$C_\infty = \int_\Gamma h \cdot T_\infty d\Gamma = \sum_{j=1}^{nb_2} h_{ij} A_{ij} T_{r\infty j} \quad - \quad \text{convection vector} \tag{15.15}$$

$$R_\infty = \int_\Gamma \varepsilon \sigma T_{r\infty}^4 dT = \sum_{j=1}^{nb_3} \varepsilon_{ij} \sigma A_{ij} T_{r\infty j}^4 \quad - \quad \text{radiation vector} \tag{15.16}$$

The last four equations of the above are correct only if there is an external radiation model and it is not influencing itself. With such a situation then these four equations can be merged to one equation which describes the exchange of radiation among surface patch elements.

$$RT^4 - R_\infty = \int_\Gamma q_r \Gamma = \sum_{j=1}^{nb_3} q_{rj} A_{ij} = R_A q_r \tag{15.17}$$

$$\sum_{j=1}^{N} \left(\frac{\delta_{ij}}{\varepsilon_j} - F_{ij} \frac{1 - \varepsilon_j}{\varepsilon_j} \right) q_{rj} - \sum_{j=1}^{N} (\delta_{ij} - F_{ij}) \sigma T_j^4 = -F_i \sigma T_{r\infty}^4 \tag{15.18}$$

This can be rewritten into matrix form:

$$R_B - q_r + R_C T^4 = R_{B\infty} \tag{15.19}$$

Equations 15.17, 15.18 expand the current equation system of one equation and one unknown to two equations and two unknowns.

$$M\dot{T} + KT + CT + R_A q_r = Q + C_\infty \tag{15.20}$$

$$R_B - q_r + R_C T^4 = R_{B\infty} \tag{15.21}$$

Non-linear solver

The Newton-Raphson method is used as the non-linear solver which uses the following equation system. This method is based on incomplete Taylor's expansion. This expansion is used in the equation 36.

$$\begin{aligned}
N(d) &= F && (15.22)\\
N(d^i + \Delta d^i) &\simeq F && (15.23)\\
N(d^i + \Delta d^i) &= N(d^i) + \frac{\partial N}{\partial d}|_{d^i}\Delta d^i && (15.24)\\
\frac{\partial N}{\partial d}|_{d^i}\Delta d^i &= F - N(d^i) && (15.25)\\
K_T(d^i)\Delta d^i &= F - N(d^i) && (15.26)
\end{aligned}$$

with the solution updated using:

$$d^{i+1} = d^i + \Delta d^i \tag{15.27}$$

Conduction

The surface conduction model allows for the radiation solver to calculate the conduction between neighbouring super patches based on their defined thickness and thermal properties. Each boundary set and therefore super patch has a defined thickness so that along with the super patch area the super patch mass is then known. The planar conduction is solved as a 2-dimensional heat transfer problem with a uniform thickness for each boundary which implies that the surface conduction represents thin components such as heat shields.

Surface Tangential Conduction Matrix

The conduction matrix can be expressed as:

$$KT = \int_\Gamma k\frac{T - T_n}{r} d\Gamma = \sum_{j=1}^{nb_j} k_i \frac{A_{ij}}{r_{ij}}(T_i - T_j) \tag{15.28}$$

where

A_{ij}	is the absolue distance of line $C_pD_p = \lvert\mathbf{r}_{C_pD_p}\rvert$
r_{ij}	is the absolue distance of line $AB = \lvert\mathbf{r}_{AB}\rvert$
k_i	is the conductivity
T_i, T_j	are the temperatures
i	represents the i-th super-patch
j	represents the j-th neighbour super-patch

An example of the surface conduction model results for a simple test where one patch is hotter than its neighbours is shown below.

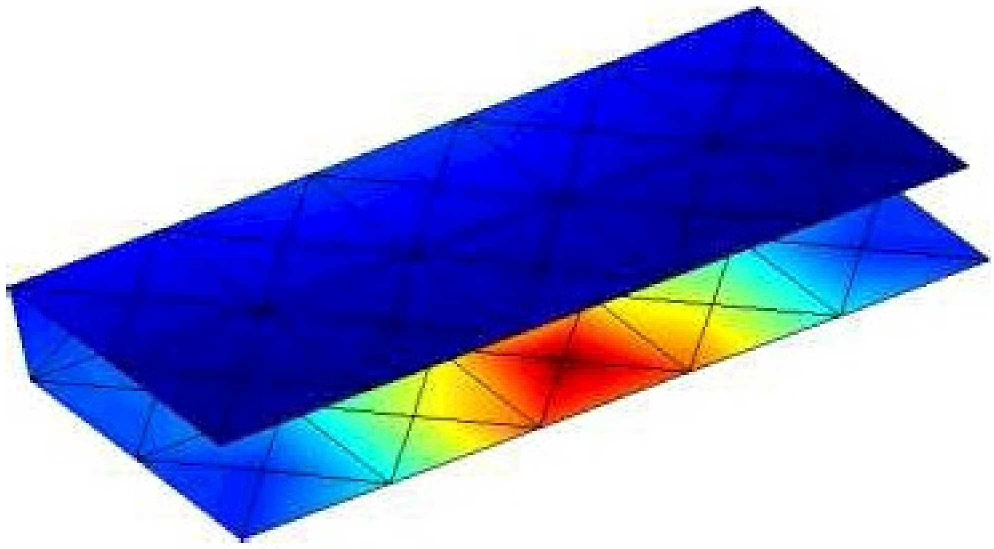


Figure 15.10: Example results for the surface conduction model showing the patch temperature

16 MODELLING FANS

16.1 Introduction And Overview

There are currently two fan models implemented within VECTIS-MAX . These are "Subdomain" and "1D". Both models require the specification of a pair of inlet/outlet boundaries/subdomain-interfaces. No conventional I/O data should be entered for these boundaries; instead, a new fan should be created as shown in Fig (16.1).

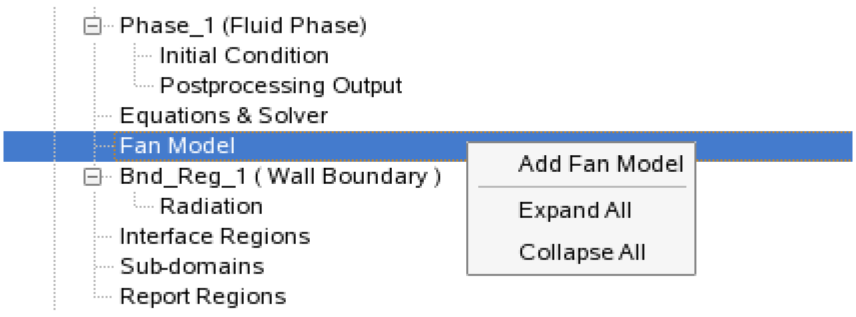


Figure 16.1: Adding a fan.

In general, the fan geometry can be simplistic and so the whole fan assembly can be represented by a simple model geometry (e.g. cylinder shape) which has the appropriate inlet and outlet areas that correspond to those of the real fan geometry.

Fan parameters

The various fan panel parameters in Fig (16.2) are explained below. Some of these are only relevent to one of the model types (as indicated).

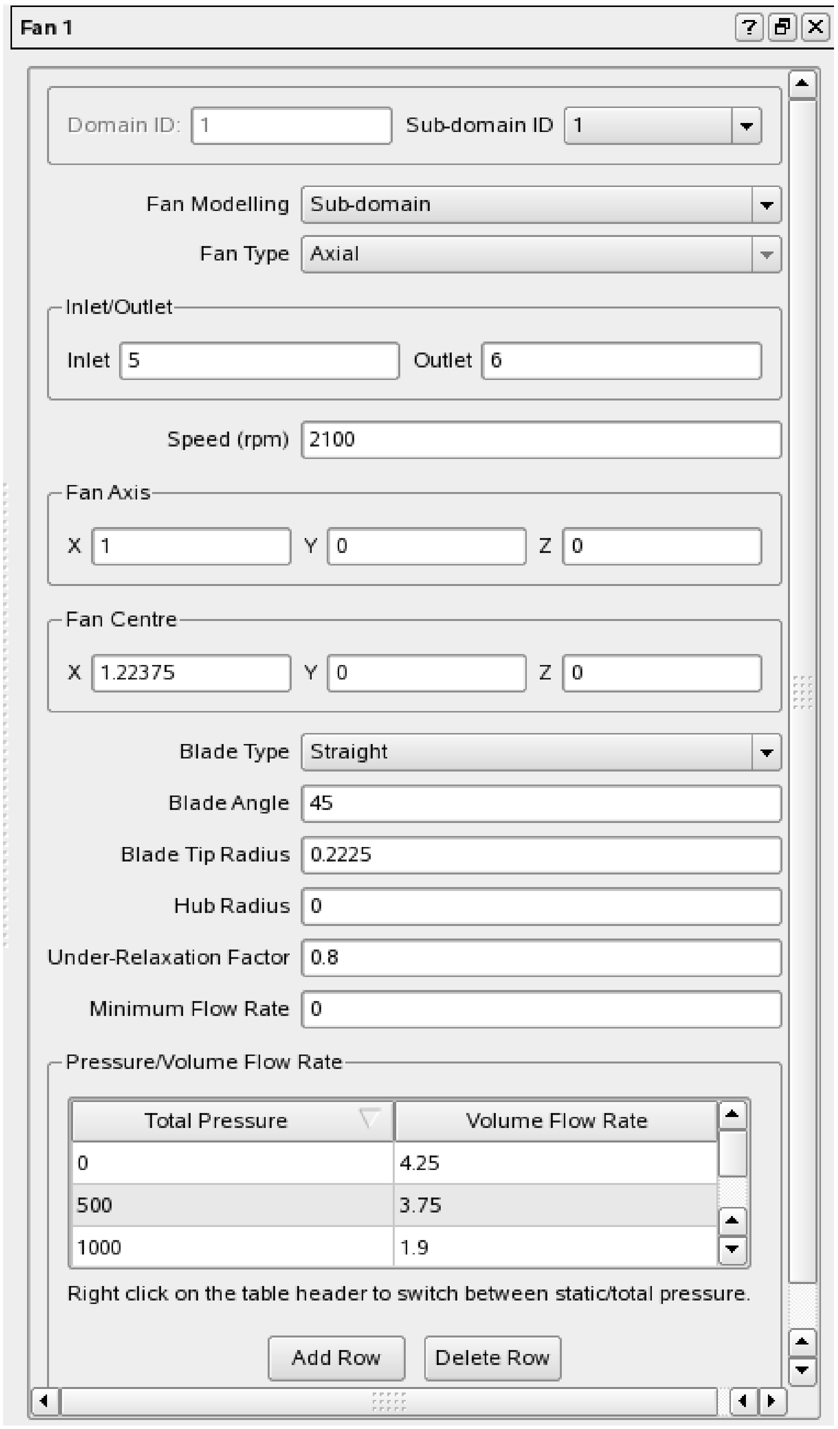
Subdomain ID

This is relevant when the "Fan Modelling" is set to "Subdomain". It specifies which subdomain is designated as a fan.

Fan Modelling

This is either "Subdomain" or "1D".

Fan Type



Domain ID: 1 Sub-domain ID 1

Fan Modelling: Sub-domain

Fan Type: Axial

Inlet/Outlet

Inlet 5 Outlet 6

Speed (rpm) 2100

Fan Axis

X 1 Y 0 Z 0

Fan Centre

X 1.22375 Y 0 Z 0

Blade Type: Straight

Blade Angle: 45

Blade Tip Radius: 0.2225

Hub Radius: 0

Under-Relaxation Factor: 0.8

Minimum Flow Rate: 0

Pressure/Volume Flow Rate

Total Pressure	Volume Flow Rate
0	4.25
500	3.75
1000	1.9

Right click on the table header to switch between static/total pressure.

Figure 16.2: Fan panel.

This is either Axial or Radial (1D Fan Model ONLY).

Inlet/Outlet

These are the boundary or subdomain-interface IDs depending on the model type selected.

Speed

This is the fan speed (RPM).

Fan Axis

This is the fan's axis. In the case of an axial fan this will correspond to the inlet & outlet axis. For radial fans, this will define the inlet axis.

Fan Centre

This is the fan's centre.

Blade Type

This is either "Straight" or "Twisted" (1D model ONLY).

Blade Angle (Subdomain model ONLY)

This is the angle the blade makes with the fan's axis .

Blade Tip Radius

This is the radial extent of the fan.

Hub Radius

This is the radius of the fan's hub.

Under-relaxation Factor

This is the under-relaxation factor.

Minimum Flow Rate (Subdomain model ONLY)

This is sets a lower limit on fan's flow rate.

Pressure/Volume Flow Rate

This is a table used to store the fan's characteristic curve. It should be a monotonically decreasing curve (i.e. $Pressure \downarrow$ as $VolumeFlow \uparrow$).

16.2 Subdomain model

Selecting this model assumes that a subdomain has already been defined in the corresponding mesh file. This model is currently only applicable to axial fans with straight blades. It is also assumed that the inlet flow velocity is uniform and has no tangential component. For a constant fluid density throughout the fan, the axial flow velocity (V_axial) at inlet must equate to that at the outlet (as enforced by mass conservation). The fan is assumed to impart a tangential force (F_θ) on the fluid along its blades, and is directly related to the change in tangential velocity between the fan's inlet/outlet (Eq (16.1)).

$$F_\theta = \rho_{fan} V_{axial} A \left[V_\theta^{outlet} - V_\theta^{inlet} \right] \tag{16.1}$$

As the inlet flow is assumed to be purely axial, V_θ^{inlet} is taken to be zero. For V_θ^{outlet}, if the fluid is assumed to leave along the blade surface (in the blade's frame of reference) and the axial component is V_{axial}, then the relative tangential velocity is $V_{axial}\tan(\beta)$ (see Fig (16.3).

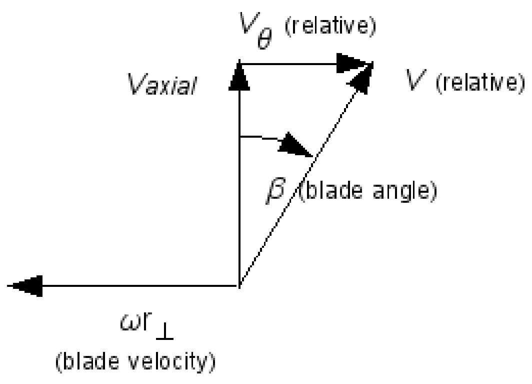


Figure 16.3: Velocity vectors at fan's outlet.

Thus, the absolute outlet tangential velocity is:

$$V_\theta^{outlet} = \omega r_\perp - V_{axial}\tan(\beta) \tag{16.2}$$

where ω is the fan's rotational frequency (rad/s), $r_\perp$ is the prepندicular distance from the fan's axis. β is the angle between the blade's normal and its direction of motion as shown in Fig (16.4). Eq (16.2) can then be plugged into Eq (16.1) to give:

$$F_\theta = \rho_{fan} V_{axial} A \left[\omega r_\perp - V_{axial}\tan(\beta)\right] \tag{16.3}$$

where ρ_{fan} is the average fan density. Assuming the total force to act along the blade's normal, we can calculate the axial component as shown in Eq (16.4).

$$F_{axial} = F_\theta tan(\beta) \tag{16.4}$$

The cartesian forces, $F_{x,y}$, experienced by a given point (in the plane of the fan) can be resolved by using its radial position, $r_\perp$, and corresponding x-y components (x_{dist}, y_{dist}), see Eq (16.5)) and Fig (16.5)).

$$\begin{aligned} F_x &= -F_\theta \frac{y_{dist}}{r_\perp} \\ F_y &= -F_\theta \frac{x_{dist}}{r_\perp} \end{aligned} \tag{16.5}$$

These linear forces are then assimilated as momentum sources. In addition, these forces are scaled by a "source factor" in such a way that the fan's operating point is eventually found.

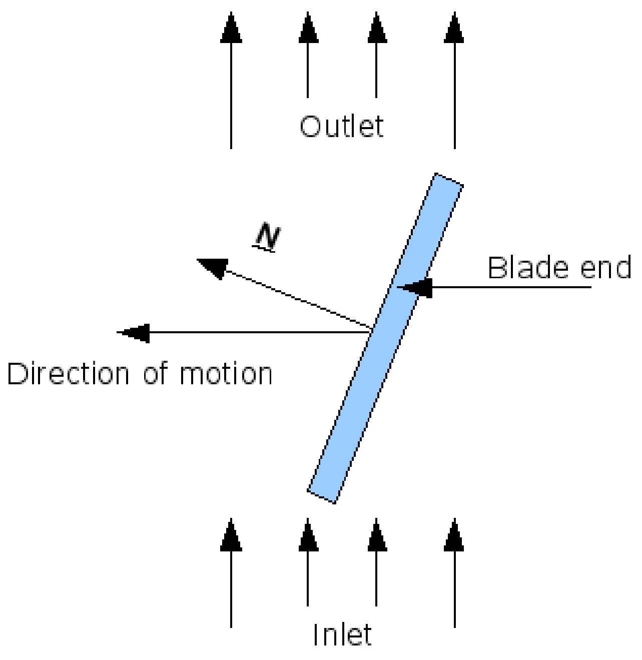


Figure 16.4: Top view of an axial fan.

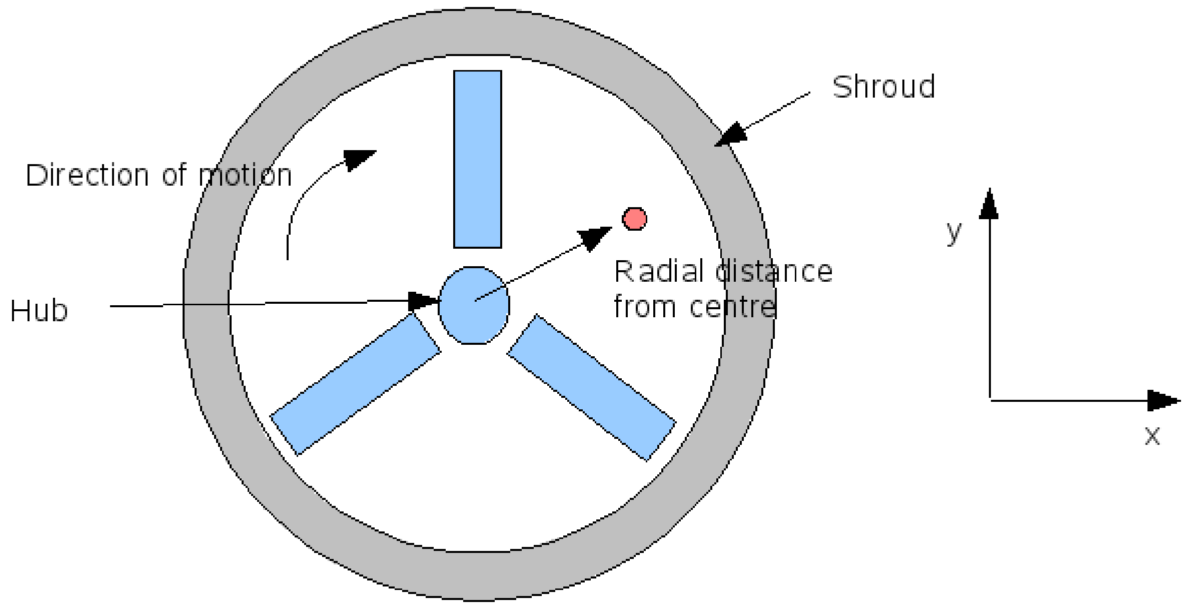


Figure 16.5: Front view of an axial fan (shrouded).

16.3 1D model

The model represents a fan using a pair of coupled inlet/outlet boundaries, so that the body of the fan itself lies outside the VECTIS flow domain, and the flow generated by the fan is imparted via the inlet/outlets. The model can be used to simulate axial fans, in which the flow enters and leaves parallel to the direction of the fan's axis, or radial fans, in which the flow enters parallel to the axis, but leaves in a radial direction.

The total pressure developed by a fan is dependent on the fluid volumetric flow rate. In general, as the flow rate decreases, the fan's total pressure (and static) will increase. The fan's static pressure will reach a maximum value as the flow rate falls to zero. Conversely, at the other extreme, the flow rate will reach a maximum as the fan's static pressure falls to zero. The relationship between the fan's pressure and volumetric flow rate is known as its characteristic, which is normally provided by the fan manufacturer (see Fig (16.6).

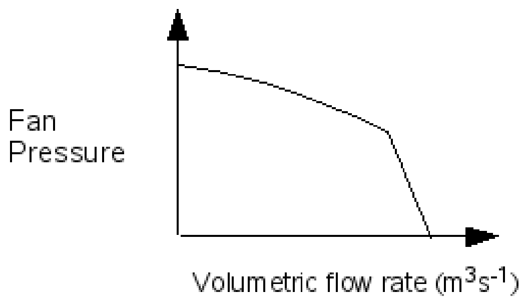


Figure 16.6: Typical fan characteristic curve (monotonic).

The fans's characteristic curve can be used during a CFD simulation to look-up the flow rate for a given fan pressure. The fan pressure is the pressure rise between the inlet and outlet. From the flow rate the mean axial velocity can be determined. For the mean tangential velocity component, we can use the "Euler Equation" for turbo-machinery (Eq (16.6)).

$$\Delta P = \rho\omega r_{\perp} v_{\perp}(outlet) - \rho\omega r_{\perp} v_{\perp}(inlet) \tag{16.6}$$

where ΔP is the total pressure rise, $\omega r_{\perp}$ is the impeller (blade) velocity and $v_{\perp}$ is the fluid tangential velocity.

Straight Blade

Here, the tangential and axial velocities are assumed to increase linearly with radial distance from the fan centre. As a consequence, the outflow is characterised by a constant flow angle.

Twisted Blade

For this blade design, the tangential velocity is assumed to vary inversely with radial distance from the fan centre. The axial velocity field is assumed to be uniform.

Axial Fans

Axial fans are represented by two boundaries which are normal to the fan's axis (see Fig (16.7)). No boundary conditions need to supplied elsewhere (i.e. from within the boundary panel) for these boundaries.

Radial Fans

The figure below (Fig (16.8)) illustrates a the geometric representation of the radial model.

For radial fans, uniform velocity distributions are assumed at the inlet and outlet. The axial component of the flow at the outlet is determined from the "Outlet Flow Vector" as specified in the fan

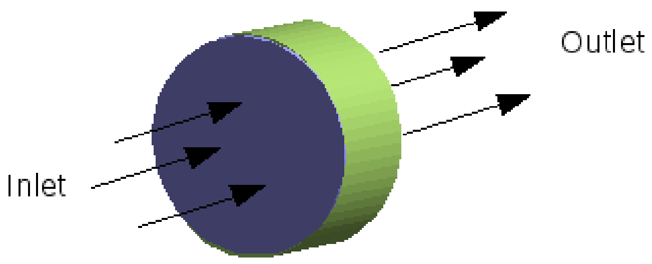


Figure 16.7: Axial fan representation.

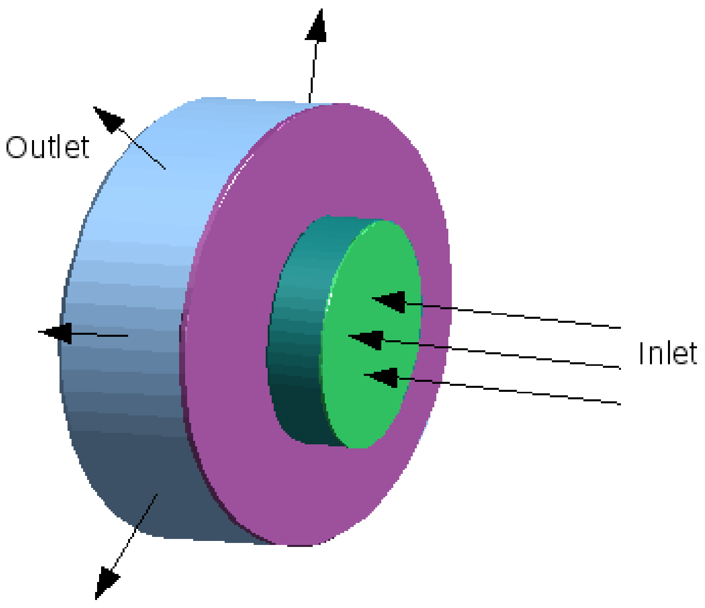
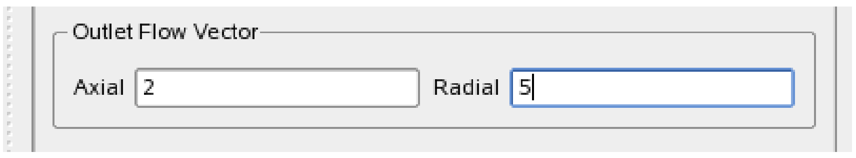


Figure 16.8: Radial fan representation.

panel (see Fig (16.9)).



Outlet Flow Vector

Axial	2	Radial	5

Figure 16.9: Outflow vector specification for radial fan setup.

17 USING SOLVER

17.1 Introduction

The setting up and running of the solver is explained more fully within this chapter. As illustrated before, this can all be done from within *R–Desk* using the Solver Setup Tree available in a VECTIS project. The input file for the solver is a formatted text file containing information about solution control, boundary conditions, initial conditions etc. All physical units in the file and throughout VECTIS are SI units (kg, m, s, K). There are no exceptions to this.

The contents of the solver setup input panel changes dynamically according to the selected branch of the solver setup tree .

Once the "input file" is set up, the solver can run either from the command line or via *R–Desk* (see Section (17.15)). The command-line interface is as follows:

```
vsolve [-np #num] [-grid #id] [-debug #level] #project
```

where "#project" is the project name (base name of the "input file"). The "-np" option is used to specify the number of processors for a parallel computation. The "-grid" option is used to specify grid file format. By default, this is assumed to be native (#id = 0).

ID	TYPE
0	Native (GRD)
1	Universal (unv)
2	Vectis-3 (DAT)
3	Nastran (nas)
4	Star CCM+ (ccm)
5	Spider (flma)
6	CGNS (cgns)

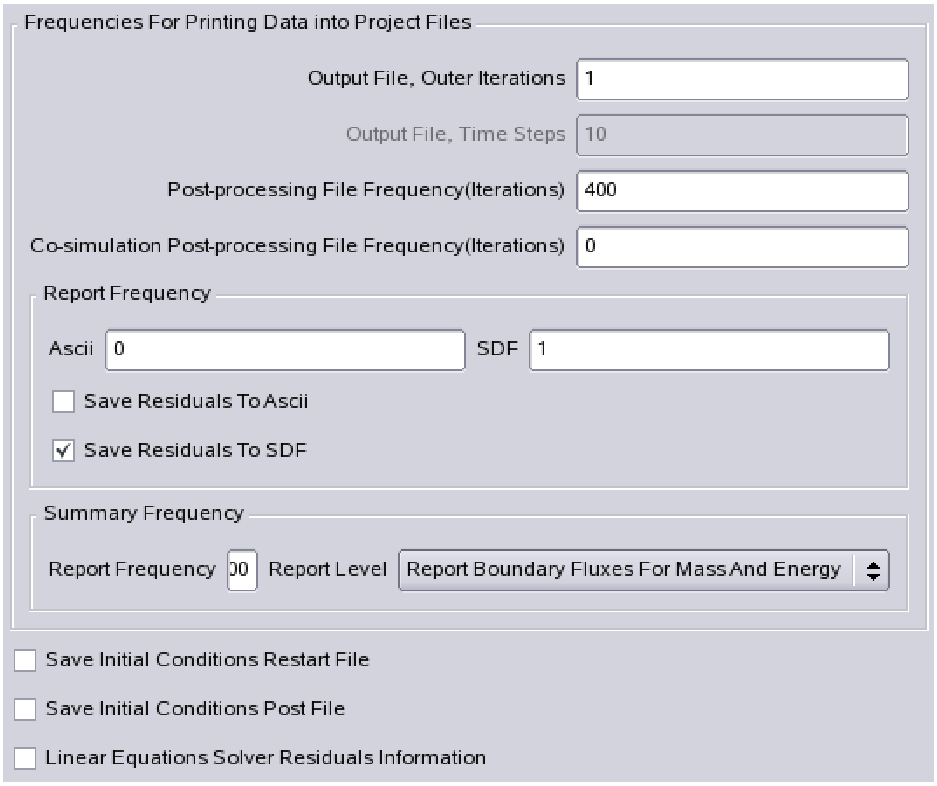
Table 17.1: Grid types supported.

The "-debug" option is to used help debug any problems that may occur. The higher the #level, the higher the verbosity of output to the screen.

17.2 Global Domain

The "Global Domain" panel is at the top level of the "Solver Setup Tree". From within here the parameters that can be controlled are:

Input Mesh Filename (see Figure (7.2))
Time Mode (see Figure (14.2))
File Output Frequencies


Frequencies For Printing Data into Project Files

Output File, Outer Iterations	1
Output File, Time Steps	10
Post-processing File Frequency(Iterations)	400
Co-simulation Post-processing File Frequency(Iterations)	0

Report Frequency

Ascii 0 SDF 1

☐ Save Residuals To Ascii

☑ Save Residuals To SDF

Summary Frequency

Report Frequency)0 Report Level Report Boundary Fluxes For Mass And Energy

☐ Save Initial Conditions Restart File

☐ Save Initial Conditions Post File

☐ Linear Equations Solver Residuals Information

Figure 17.1: GroupBox for "Frequencies For Printing Data into Project Files".

Referring to figure (17.1), Output File, Outer Iterations controls the outer iteration output frequency to screen and "out" file. Output File, Time Steps sets the output frequency for unsteady runs. Post-processing File Frequency sets the frequency to write to the "post" files. Depending on the Time Base chosen, this frequency changes its meaning: for Non-dimenensional , it refers to the frequency (per cycle), for 2-Stroke / 4-Stroke or Seconds , it is an interval in degrees or seconds respectively. Report Frequency controls the frequency to write to the "rep" (SDF) file and the equivalent ASCII files. By default, the SDF frequency is set to 1 and the ASCII frequency is set to zero (off). The ASCII output files are explained in Section (17.6). Summary Frequency controls the frequency of writing run summaries to screen and "out" file via the Report Frequency LineEdit. Report Level controls amount of information contained therein.

Co-simulation Post-processing... LineEdit is used to control the output frequency of variables when VECTIS is coupled with WAVE (see Sec (17.16)).
Save Initial Conditions... CheckBoxes are used save the initial state of the system to the "Restart"

0	Off
1	Report CPU timings
2	Report boundary fluxes for mass and energy.

Table 17.2: Report levels for "Summary Frequency" GroupBox.

and/or "Post" file. The initial conditions are saved after any flow potential calculations are performed but before the first time-step/iteration. The Linear Equations Solver Residuals Information provides extra residual information for each equation and each inner iteration of the solver. This will write to the screen and output files in the format shown below.

```
Domain: 1 FI: U RES0 = 2.466729E+00
Domain: 1 FI: U ITER = 1 RES = 3.730890E-03
Domain: 1 FI: V RES0 = 5.618043E-01
Domain: 1 FI: V ITER = 1 RES = 5.258057E-03
Domain: 1 FI: W RES0 = 6.936036E-02
Domain: 1 FI: W ITER = 1 RES = 2.324654E-03
Domain: 1 FI: P RES0 = 8.562884E-03
Domain: 1 FI: P ITER = 1 RES = 8.590540E-01
Domain: 1 FI: P ITER = 2 RES = 8.876634E-01
Domain: 1 FI: P ITER = 3 RES = 9.081014E-01
```

17.3 Restart Control

During the calculation, restart files are written at a user defined frequency throughout the calculation. Restart files are written with an interval specified in the 'Restart File Frequency' field.
Two restart files are generated, projectname.rst0_runnumber and projectname.rst1_runnumber are written alternately every restart frequency time steps.

The "Restart Control" panel controls the restart file reading and writing. Shown below is the restart reading GroupBox:

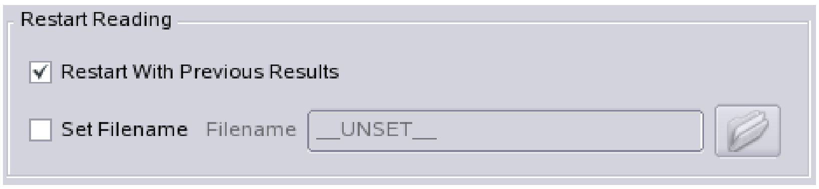
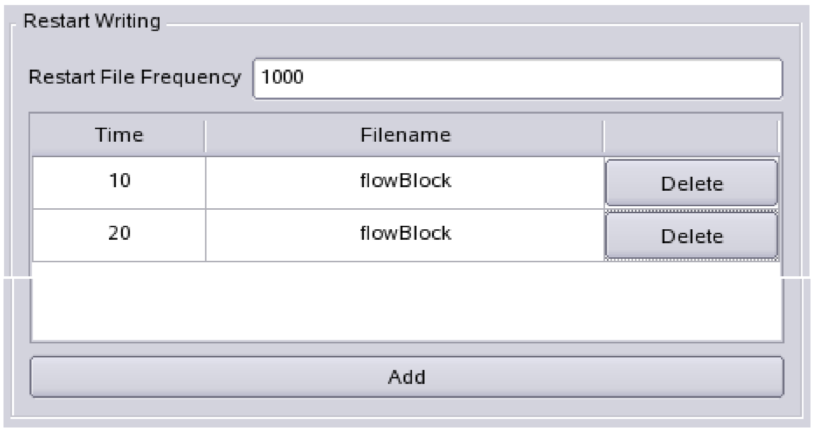


Figure 17.2: GroupBox for "Restart Reading".

Restart reading is activated by clicking on the Restart With Previous Results . This will instruct the solver to restart from the newest "rst" files (with the same project base name). To override this behaviour, the user can activate the Set Filename CheckBox and enter a specific filename in the Filename LineEdit. The ideal restart frequency is a compromise between how far back in the calculation you will have to go to restart in the case of the calculation crashing or system failure and how much time the calculation spends writing the restart data to disk. If a very short frequency

is chosen then it can reduce the overall speed of the calculation.
The Restart File Frequency LineEdit in the Restart Writing GroupBox specifies how often to write to the restart ("rst") files. In addition, the user can add specific restart files to be written at certain times by clicking on the Add button and entering the time and corresponding filename. The filename is exactly as specified here; no project name or run number are added for timed restart files. See figure Fig (17.3).



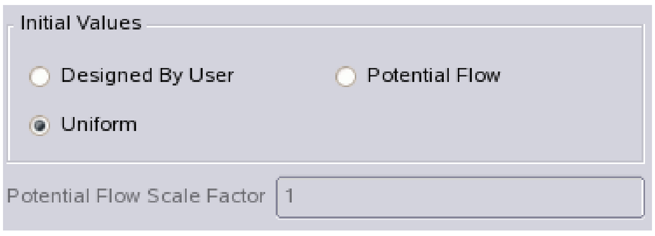
Restart File Frequency: 1000

Time	Filename
10	flowBlock
20	flowBlock

Figure 17.3: GroupBox for "Restart Writing".

17.4 Fluid Domain

The fluid domain panel defines domain/material name, material id and multiphase modelling type (see Figure (8.1)). Initial values can be set as Uniform, User Defined or "Potential Flow". For "User Defined", the user must write and compile a user function (see Section 18.4.2). The "Potential Flow" option uses the solution from a simplified model (irrotational flow assumed) which is initially run prior to the main solver. This is normally very quick. If necessary, the user can scale down (between 0-1) the estimated inflow velocities via the Potential Flow Scale Factor LineEdit.



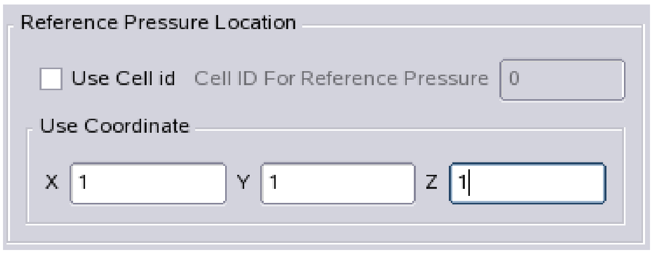
Designed By User
Potential Flow
[✓ Uniform]
Potential Flow Scale Factor: 1

Figure 17.4: GroupBox for "Initial Values".

The Reference Pressure Location GroupBox is for setting the reference pressure cell or, more conveniently, the xyz-coordinate. See below.

Other reference quantities that can be set for this fluid domain are shown in Fig (17.6).

Setting of Body Force options has been described in the section dealing with modelling buoyancy–



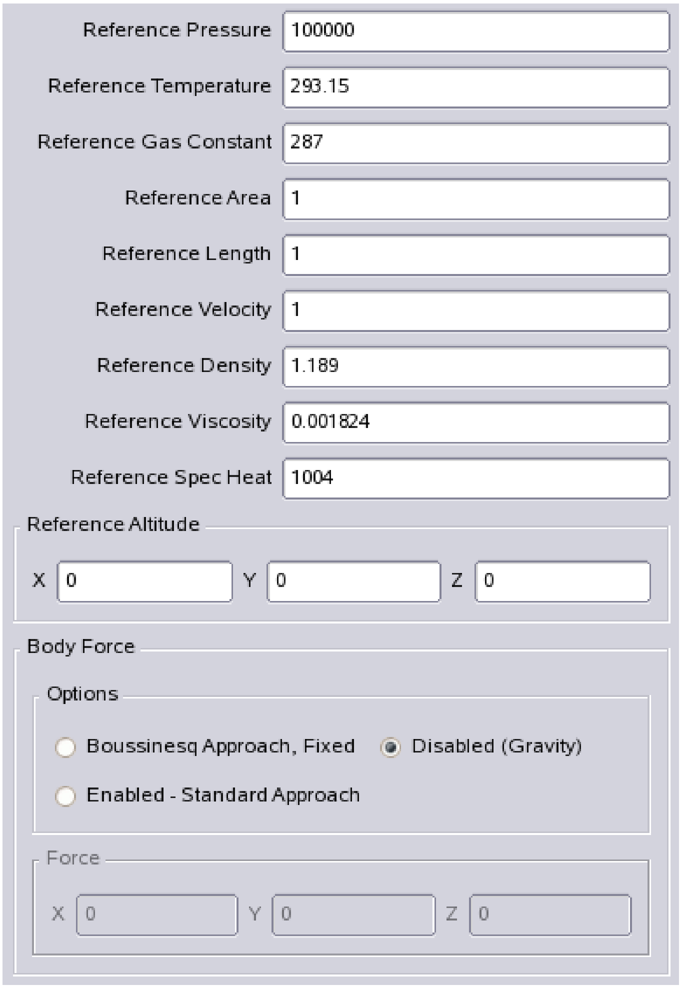
Reference Pressure Location

Use Cell id	☐
Cell ID For Reference Pressure	0

Use Coordinate

X	1
Y	1
Z	1

Figure 17.5: GroupBox for "Pressure Monitoring".



Reference Pressure	100000
Reference Temperature	293.15
Reference Gas Constant	287
Reference Area	1
Reference Length	1
Reference Velocity	1
Reference Density	1.189
Reference Viscosity	0.001824
Reference Spec Heat	1004

Reference Altitude

X	0
Y	0
Z	0

Body Force

Options

Boussinesq Approach, Fixed [✓ Disabled (Gravity)] Enabled - Standard Approach

Force

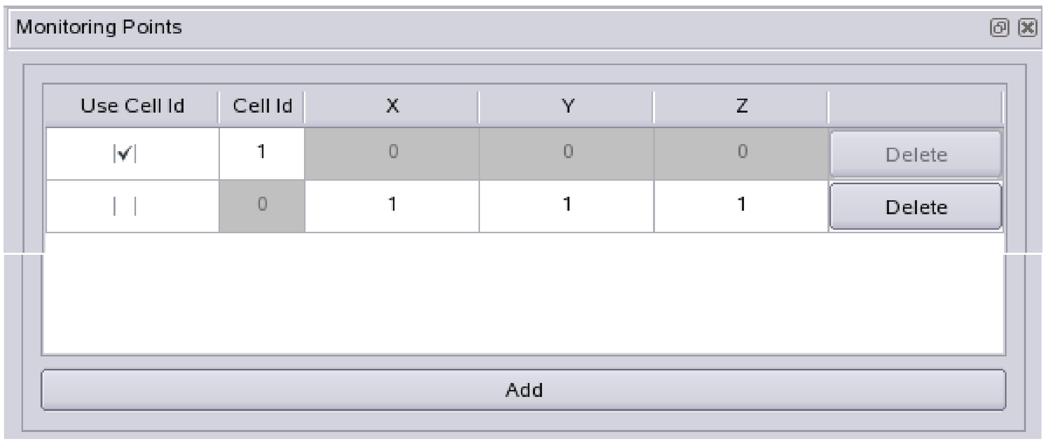
X	0
Y	0
Z	0

Figure 17.6: Reference values LineEdit's.

driven flows, Figure 10.8.

17.5 Monitoring Points

This panel is used to monitor various flow quantities (velocity, pressure etc.) at specific locations. Either the cell Id or xyz-location can be used. Monitoring values are written to the "mon" files. The first monitoring point is also written to the screen and main "out" file.



Monitoring Points

Use Cell Id	Cell Id	X	Y	Z	
✓	1	0	0	0	Delete
	0	1	1	1	Delete

Figure 17.7: Monitoring Points Panel.

17.6 File Output

Here the frequency of data reporting output is chosen. The monitoring and convergence data written to the screen is also written to the .out file at the intervals specified. If zero values are used then the output is not written.

The frequency of post-processing file writing is specified in the 'Post-processing File Frequency' box.Additional output files can be written for monitoring data, boundary data, report regions and domain data. These can be either in ASCII column format or Ricardo SDF binary format. The frequency of the two types of file can be chosen independently by entering the required intervals in to the 'Report Frequency' boxes. Again an interval of zero will mean that the data is not written.

The ASCII files include header rows detailing the data in each of the columns. Separate files will be written for the different data types. Each file adheres to the format:

```
projectname.<type>_<runnumber>
```

Where `<type>` is given by :

Domain data	dom
Monitoring data	mon
IO data	io
Wall data	wall
Arbitrary Surface	surf

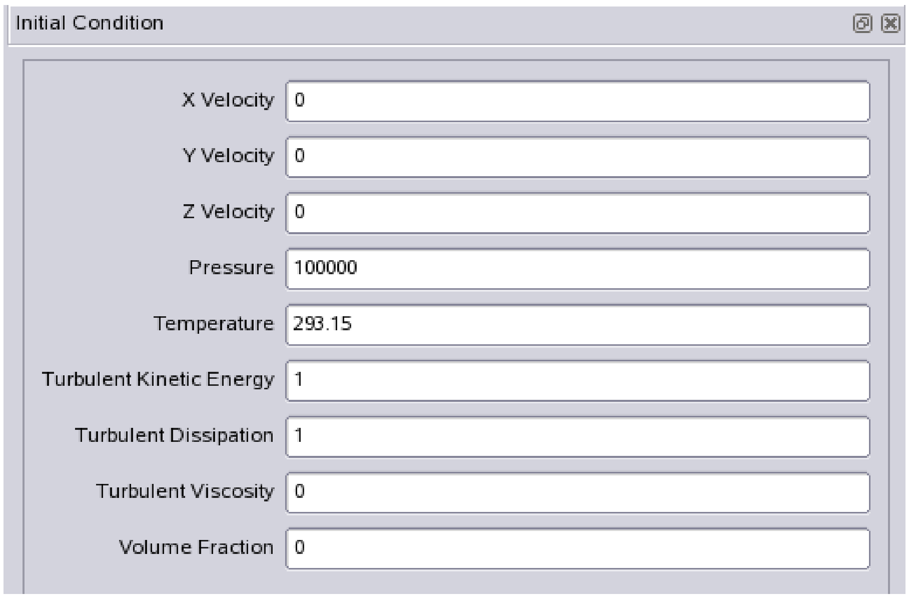
Table 17.3: Naming convention for different output file types.

The ASCII data files are used by the 'Live Update' utility in *R–Desk* .

The SDF binary format is written to a single report file named projectname.rep_runnumber. It contains the data for all the additional output. It can be viewed and plotted in *R–Desk* using the 'SDF File Manager' and 'XY Plot Manager'

17.7 Fluid Phase

The main panel was described in Section (8.6.2). The subpanel Initial Condition , shown below, lets the user specify initial flow velocity, pressure, temperature etc. The Postprocessing Output subpanel is for specifying which quantities are to be written to the postprocessing file ("post"). See Figure (17.9).



X Velocity	0
Y Velocity	0
Z Velocity	0
Pressure	100000
Temperature	293.15
Turbulent Kinetic Energy	1
Turbulent Dissipation	1
Turbulent Viscosity	0
Volume Fraction	0

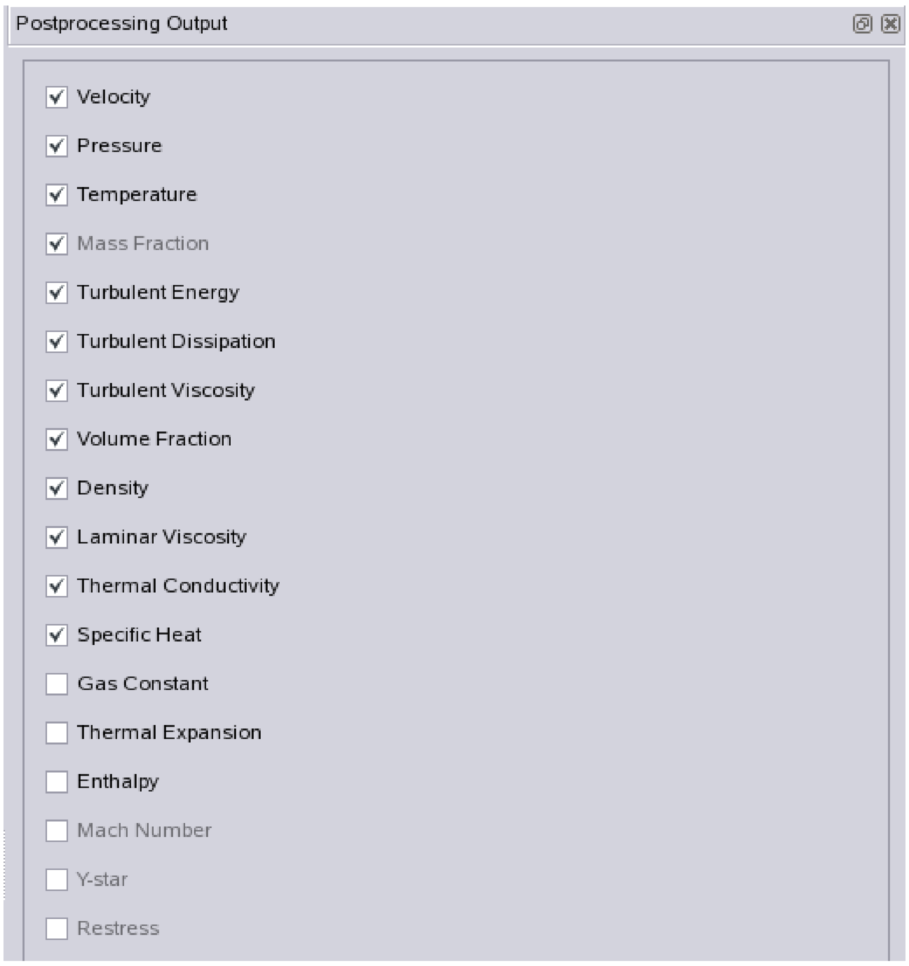
Figure 17.8: Initial Conditions Panel.

17.8 Fluid Species

The species panel (see Figure (8.9)) allows the user to add new species for given phase. It also contains subpanels Fluid Species used to define species properties, which contains the single component phase properties as well the species specific properties (see Figure (8.10)). Below the Fluid Species panel, there is Output panel used to specify the quantities to be written to the postprocessing file ("post"). See Fig. (17.10).

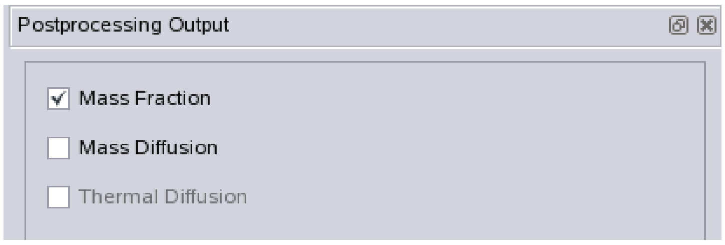
17.9 Boundary Region Definition

The Boundary Region panel was mostly explained in Section (13.1). The Boundary Report Check-Box is used to indicate whether ASCII/SDF reporting is required for this boundary. See Section 17.6 on alpha-numeric reporting for more information. Typical contents of an "io" report file are shown below:



- [✓] Velocity
- [✓] Pressure
- [✓] Temperature
- [✓] Mass Fraction
- [✓] Turbulent Energy
- [✓] Turbulent Dissipation
- [✓] Turbulent Viscosity
- [✓] Volume Fraction
- [✓] Density
- [✓] Laminar Viscosity
- [✓] Thermal Conductivity
- [✓] Specific Heat
- [ ] Gas Constant
- [ ] Thermal Expansion
- [ ] Enthalpy
- [ ] Mach Number
- [ ] Y-star
- [ ] Restress

Figure 17.9: Postprocessing Output Panel.



- [✓] Mass Fraction
- [ ] Mass Diffusion
- [ ] Thermal Diffusion

Figure 17.10: Species Output Panel.

```
!-------------------------------------------------------------------------------------------
! Iter_No   Area_TOT       Temp_AVE      Mflow_SUM     Density_AVE     S:Pres_AVE     T:Pres_AVE
!-------------------------------------------------------------------------------------------
!             m^2              K              kg/s          kg/m^3           Pa              Pa
    1      1.000000E-01  2.931500E+02 -1.188574E+00  1.188574E+00  1.008974E+05  1.009574E+05
    2      1.000000E-01  2.931500E+02 -1.188574E+00  1.188443E+00  1.006558E+05  1.007158E+05
    3      1.000000E-01  2.931500E+02 -1.188574E+00  1.188546E+00  1.002759E+05  1.003359E+05
    4      1.000000E-01  2.931500E+02 -1.188574E+00  1.188568E+00  1.000825E+05  1.001426E+05
    5      1.000000E-01  2.931500E+02 -1.188574E+00  1.188569E+00  9.999433E+04  1.000544E+05
    6      1.000000E-01  2.931500E+02 -1.188574E+00  1.188570E+00  1.000057E+05  1.000657E+05
    7      1.000000E-01  2.931500E+02 -1.188574E+00  1.188570E+00  1.000045E+05  1.000646E+05
    8      1.000000E-01  2.931500E+02 -1.188574E+00  1.188571E+00  1.000064E+05  1.000665E+05
```

The meaning of the variables are shown in Table (17.4). The "_AVE" values are area-averaged values and are calculated thus:

$$X\_AVE = \frac{1}{\sum_{i=1}^{n} A_i} \sum_{i=1}^{n} X_i A_i \tag{17.1}$$

where X_i is the value and A_i is the area at a given face i on a boundary (comprising of n faces).

Iter_No	Outer iteration number (steady state)
Time	Time (unsteady)
Tstep	Time step (unsteady)
Area_TOT	Total area
Temp_AVE	Average temperature
Mflow_SUM	Total mass flow rate
Density_AVE	Average density
S:Pres_AVE	Average static pressure
T:Pres_AVE	Average total pressure

Table 17.4: Table of all the variables contained in io files.

Similarly, "wall" report files will typically contain the following values:

Fcoeff_X	Total wall force (normalized, x-component))
Fcoeff_Y	Total wall force (normalized, y-component))
Fcoeff_Z	Total wall force (normalized, z-component))
VFcoeff_X	Viscous wall force (normalized, x-component)
VFcoeff_Y	Viscous wall force (normalized, y-component)
VFcoeff_Z	Viscous wall force (normalized, z-component)
Ypstar_MIN	y^* (minimum value over boundary)
Ypstar_MAX	y^* (maximum value over boundary)
Heat_Flux	Total heat flux
Wall_T	Average wall temperature
NearWall_T	Average near-wall temperature (over fluid cells)
Htc	Average heat transfer co-efficient

Table 17.5: Table of all the variables contained in wall files.

where yp^* is the dimensionless wall distance (turbulent, see Section (9.4.1)):

$$\frac{\rho C_\mu^{1/4} k^{1/2} yp}{\mu} \tag{17.2}$$

and yp is the perpendicular distance from the neighbouring cell centre to the wall. The wall forces are normalized as follows:

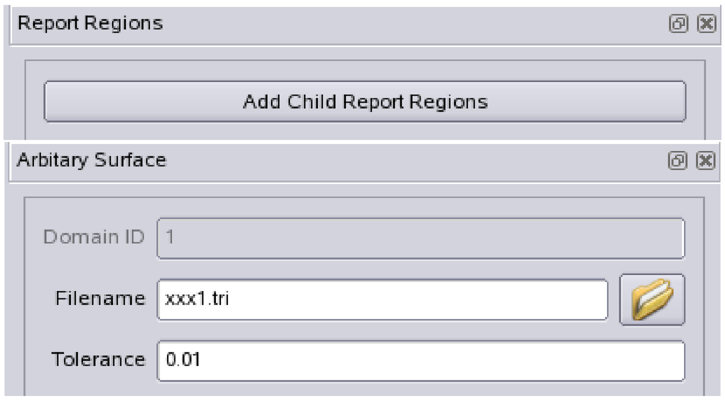
$$C_{Fi} = \frac{2F_i}{\rho_{ref} U_{ref}^2 A_{ref}} \tag{17.3}$$

see Section (13.8) for more information.

The Coupled Link Number is used to specify the WAVE link number for this boundary (see Section (17.16)).

17.10 Report Regions

The Report Regions panel, shown below, allows the user to define extra regions (other than boundary regions defined by the geometry).



Domain ID 1
Filename xxx1.tri
Tolerance 0.01

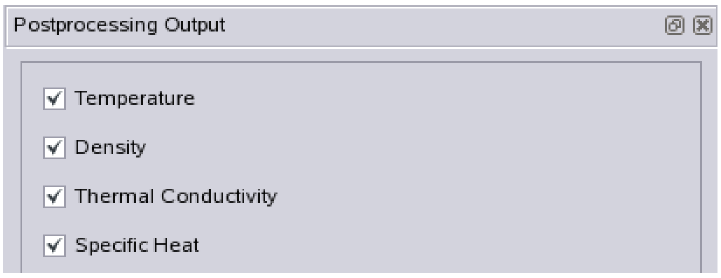
Figure 17.11: Report Regions/Arbitrary Surface Panel.

The Filename must be a VECTIS geometry (triangle) file. Mapping is done from the defined triangles to the solver mesh. Each triangle is successively divided along its longest edge until one of the following criteria are reached:

1. All nodes of a triangle are contained within the same cell.
2. All nodes of triangle are greater than, or less that the mesh extents (ie triangle is outside mesh)
3. Longest edge is less than the user specified edge Tolerance .

17.11 Solid Domain

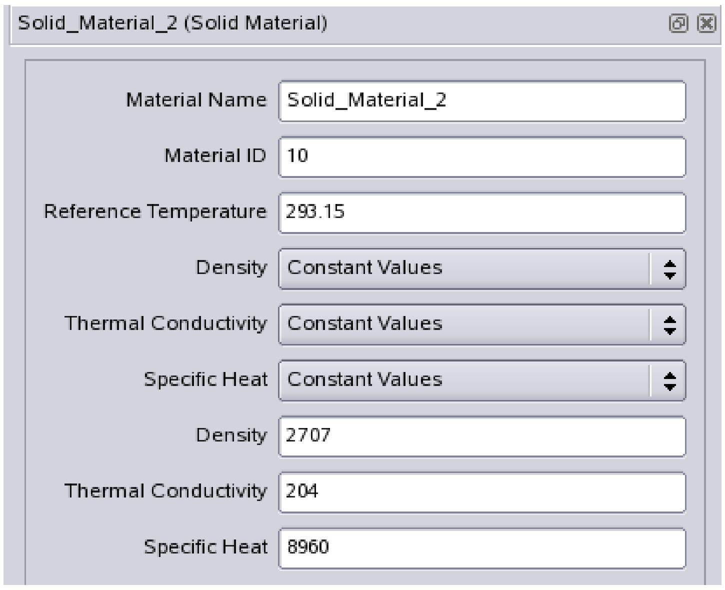
The Solid Domain panel is very similar to its Fluid Domain counterpart. Only the differences are as follows. For the Postprocessing Output panel, there are fewer available quantities that can be saved (see below).


Postprocessing Output

- ✓ Temperature
- ✓ Density
- ✓ Thermal Conductivity
- ✓ Specific Heat

Figure 17.12: Postprocessing Output Panel (Solid Domain).

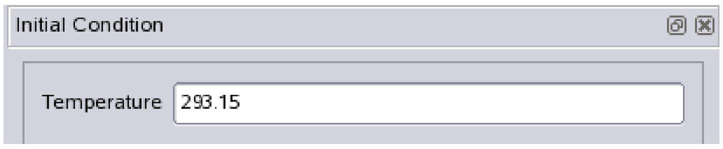
The Solid Material panel is similar to the Fluid Phase panel, except it looks like:


Solid_Material_2 (Solid Material)

Material Name	Solid_Material_2
Material ID	10
Reference Temperature	293.15
Density	Constant Values
Thermal Conductivity	Constant Values
Specific Heat	Constant Values
Density	2707
Thermal Conductivity	204
Specific Heat	8960

Figure 17.13: Solid Material Panel.

The Initial Conditions panel contains just a Temperature LineEdit.


Initial Condition

Temperature	293.15

Figure 17.14: Initial Conditions Panel (Solid Domain).

17.12 VECTIS Files

This section describes the various files used by VECTIS-MAX . All the files written out adhere to the format:

```
projectname.<type>_<runnumber>
```

where `<type>` reflects the file type, and `<runnumber>` is a 3-digit number (e.g. 005 -> run 5). The alphanumeric output files were explained in Table 17.3. Other files include the "out" file which contains screen output. For parallel runs, there are also parallel output files in each parallel sub-directory. These contain simulation preamble specific to each parallel grid file (e.g. grid size etc.). The "post" file is an SDF file which contains any postprocessing data as selected by the user in *R–Desk* . The restart files ("rst") are explained in Section (17.3). The input file read in by VECTIS-MAX has an extension "inp" without the runnumber part. However, each run automatically generates a copy of the "inp" with a runnumber attached.

17.13 Monitoring The Solution

This section describes the "Live Update" panel (Fig (17.15)) used to monitor the solution. The user is able to change the run directory to monitor, it will initially default to the current working directory. Clicking on the "Refresh Available Files" is necessary if a run has just started in the current run directory. The "Files" panel displays the averaged data for boundaries (I/O & wall) and domains. In addition, within each boundary/domain number there is a list of runs to choose from. As shown in Fig (17.16), this panel is split according to the type of averaged data.

Once the values have been chosen for a particular run, it is then necessary to open an "XY Canvas". Dragging the first value (e.g. Iter_No) from "Value" will make X-axis, the next drag-and-drop of value will make up the Y-axis. This can be repeated for subsequent graphs (on the same or different canvas) where the value-pairs are taken to be the X/Y-axis. There is no limit to the number of graphs that can be plotted on the same XY canvas. The "XY Canvas" can be renamed by right-clicking anywhere within the canvas. Figure (17.15) displays the static pressure (S:Pres_AVE) versus iteration number (Iter_No). The update interval (default 5000ms) for the XY plot can be adjusted if necessary.

17.14 Solver Control

This section describes how to control the simulation. The "Solver Control" is invoked via the "View" pull-down menu. The associated working directory can be changed if necessary. The "Run" GroupBox displays the current jobs running in this working directory. If any jobs have recently started, it may be necessary to click on the "Refresh" button. The "Solver Options" GroupBox list all the available options that can be modified. In order to apply these changes the user must then click on the "Send" button. In the Figure (17.17), run 005 is selected, and the maximum iterations

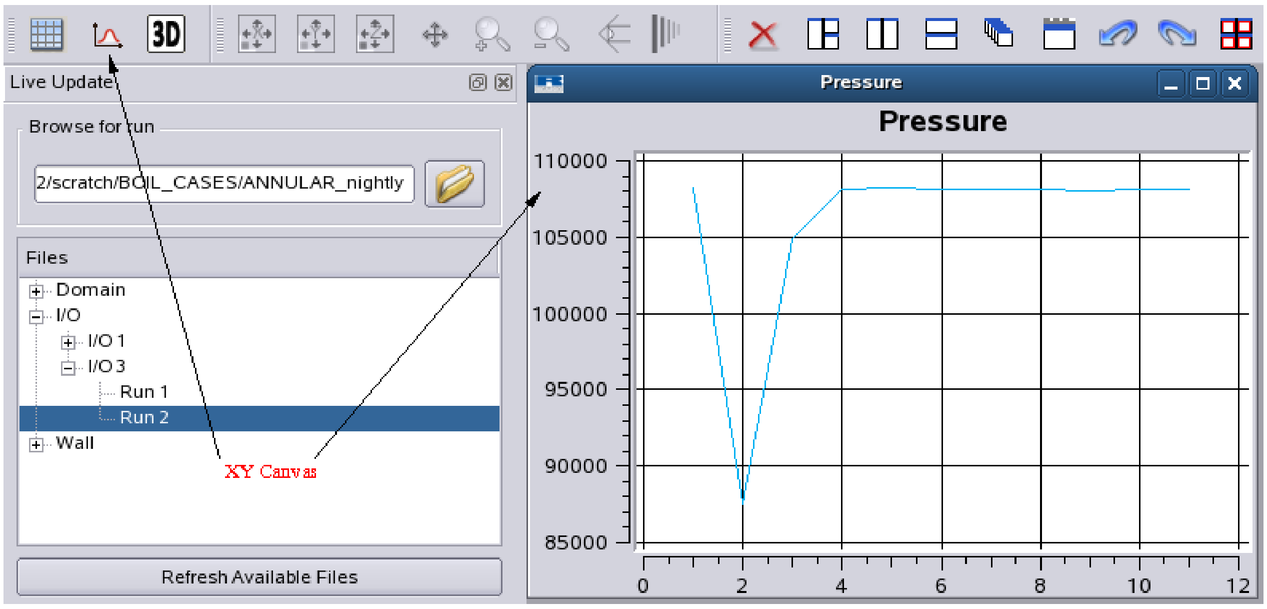


Figure 17.15: Live update & XY Canvas (pressure versus iteration number).

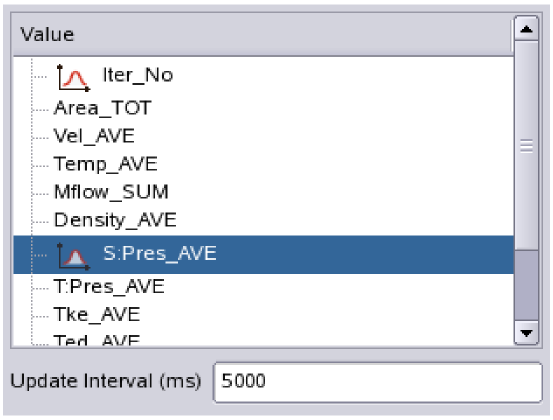
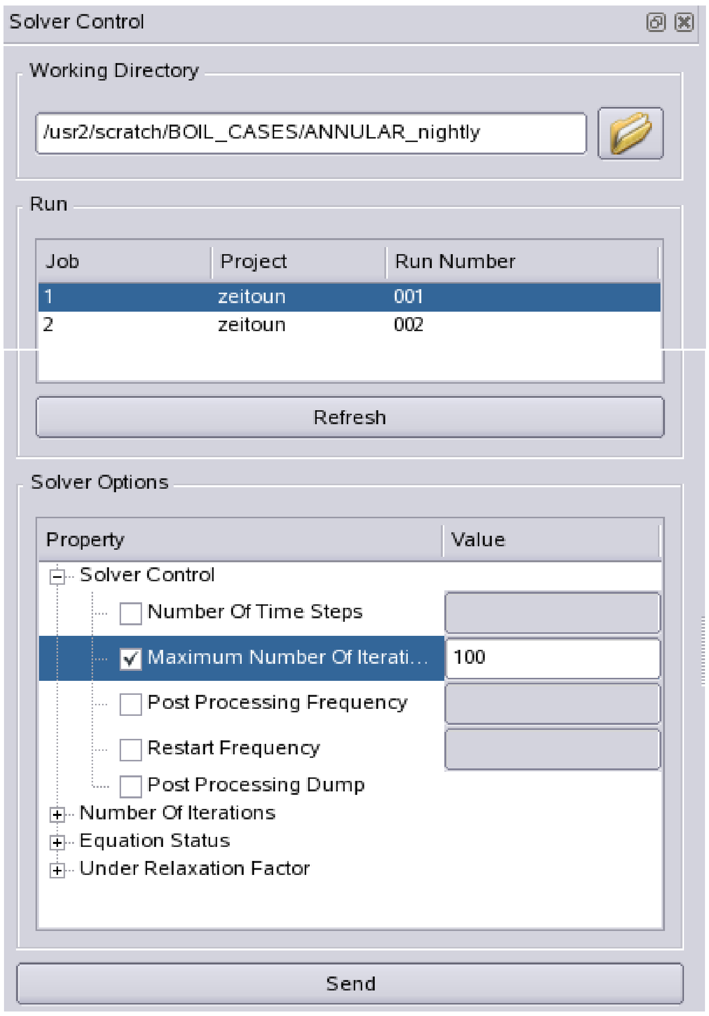


Figure 17.16: Live Update Value GroupBox.

is changed to 100. In order to stop the simulation suddenly and cleanly (with restart/postprocessing write out), it is probably simplest to set the maximum number of iterations/time steps to anything less than the current iteration/time step. The equivalent command line utility "run_control" can be used outside *R–Desk* . This program must run from the relevant working directory and is invoked without any arguments. If any jobs are currently running, the user will be prompted to change a parameter via a text menu. Once the parameter modification has finished, the user must enter "0" in order to apply these changes. See Figure (17.18).

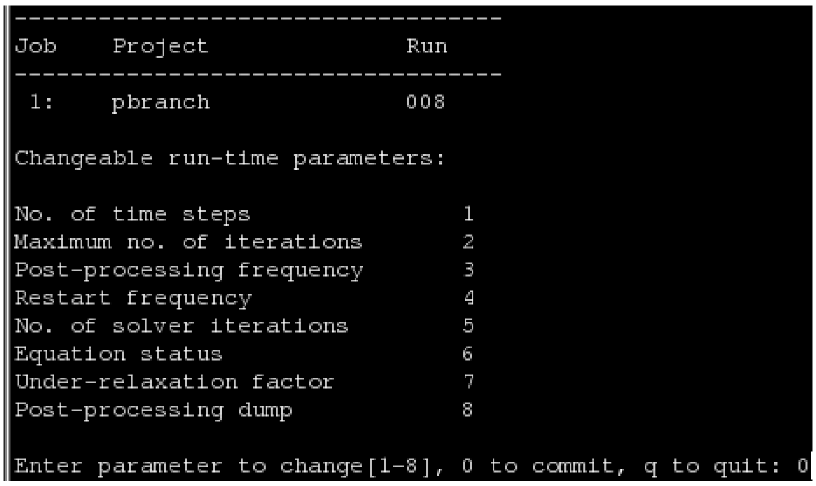


/usr2/scratch/BOIL_CASES/ANNULAR_nightly

Job	Project	Run Number
1	zeitoun	001
2	zeitoun	002

Property	Value
⊟ Solver Control	
☐ Number Of Time Steps	
☑ Maximum Number Of Iterati...	100
☐ Post Processing Frequency	
☐ Restart Frequency	
☐ Post Processing Dump	
⊞ Number Of Iterations	
⊞ Equation Status	
⊞ Under Relaxation Factor	

Figure 17.17: Solver Control panel.



```
-----------------------------------
Job     Project              Run
-----------------------------------
 1:     pbranch              008

Changeable run-time parameters:

No. of time steps                1
Maximum no. of iterations        2
Post-processing frequency        3
Restart frequency                4
No. of solver iterations         5
Equation status                  6
Under-relaxation factor          7
Post-processing dump             8

Enter parameter to change[1-8], 0 to commit, q to quit: 0
```

Figure 17.18: Solver Control via command line (run_control).

17.15 Toolchain Launcher

The mesher (vmesh), solver preprocessor (vpre) and solver (vsolve) can all be launched from within *R–Desk* . see Figure (17.19).

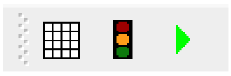
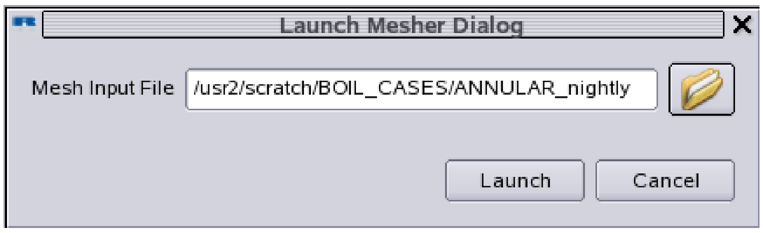


Figure 17.19: Launcher buttons, from left to right: vmesh, vpre & vsolve.

The mesher ▦ launch dialog (Fig (17.20)) allows the user to set the mesher input file.



Mesh Input File /usr2/scratch/BOIL_CASES/ANNULAR_nightly

Figure 17.20: Mesher launcher dialog box.

The vpre 🚦 launch dialog (Fig (17.21)) provides a GUI interface to the "vpre" utility. Each element corresponds to the "vpre" arguments (Sec (5.1)). The mesh file is non-optional.

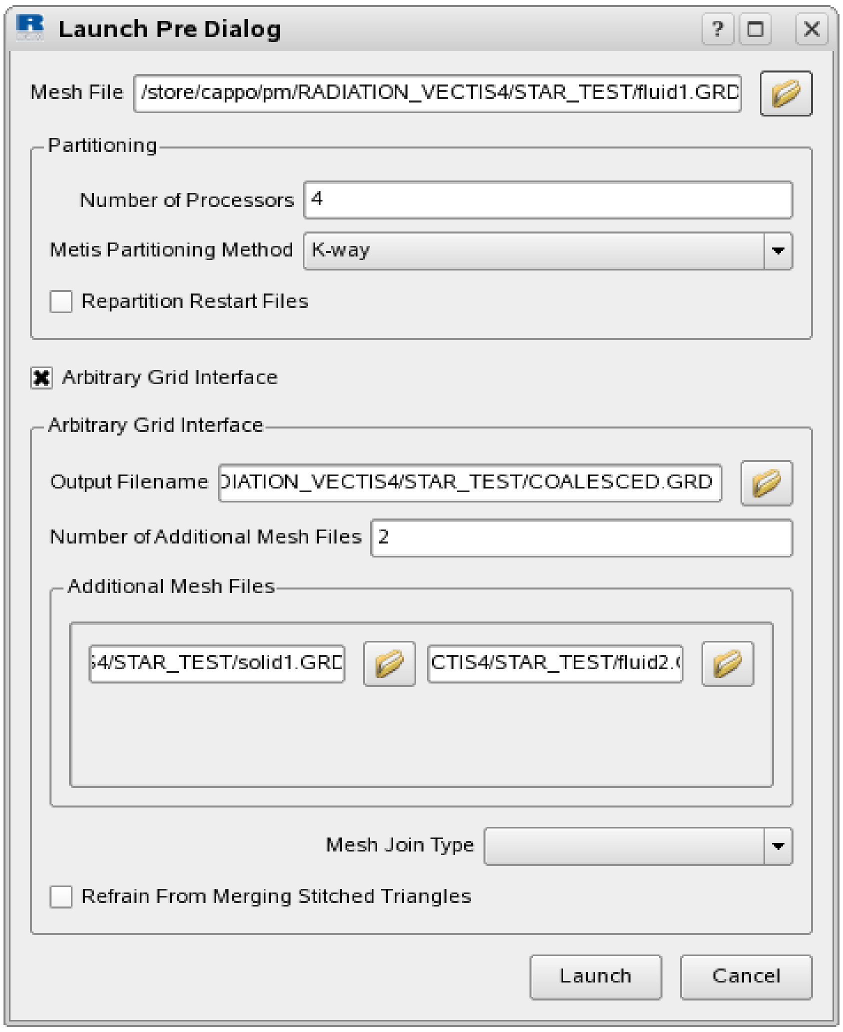
The Partitioning GroupBox contains controls for generating parallel grid files.

Control	**Vpre Option**
`Number of Processors`	-np
`Metis Partitioning Method`	-meth
`Repartition Restart Files`	-rest

Table 17.6: Partitioning controls and corresponding vpre options.

The Arbitrary Grid Interface GroupBox contains controls for joining multiple grid files (multi-domain). E.g. for joining 3 grid files together, the user would enter "2" (then `<ENTER>`) in the Number of Additional Mesh Files field and then the corresponding file names. The Mesh Join Type corresponds to command line "-meth" option.

The output file option is used to override the default output file name ("COALESCED.GRD") when joining meshes.



Mesh File: /store/cappo/pm/RADIATION_VECTIS4/STAR_TEST/fluid1.GRD

Number of Processors: 4

Metis Partitioning Method: K-way

☐ Repartition Restart Files

☒ Arbitrary Grid Interface

Output Filename: DIATION_VECTIS4/STAR_TEST/COALESCED.GRD

Number of Additional Mesh Files: 2

Additional Mesh Files: ;4/STAR_TEST/solid1.GRD, CTIS4/STAR_TEST/fluid2.(

Mesh Join Type:

☐ Refrain From Merging Stitched Triangles

Figure 17.21: Vpre launcher dialog box.

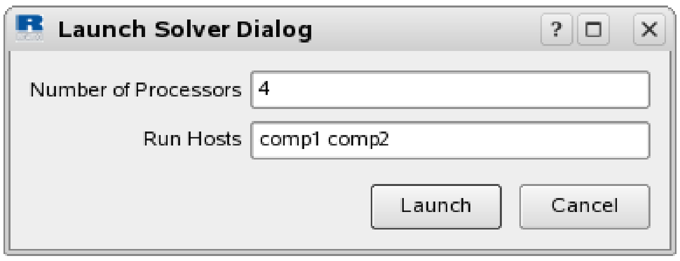
JOIN TYPE	COMMAND JTYPE NO.
`Conformal`	0
`No extrusion with meshing`	1
`Extrusion with meshing`	2

Table 17.7: Mesh join types and corresponding vpre -jtype option numbers.

The vsolve ▶ launch dialog (Fig (17.22)) allows the user to set the input file, number of processor to run on and the hosts (set to localhost if local).

17.16 WAVE-VECTIS Co-simulation

An important capability of VECTIS is the ability to couple at a time step level with the Ricardo one dimensional engine performance simulation code WAVE. WAVE-VECTIS coupling allows one or more parts of a WAVE network to be replaced by a VECTIS model. This can be useful for system



Number of Processors: 4
Run Hosts: comp1 comp2

Figure 17.22: Vsolve launcher dialog box.

components which can not be adequately represented by a one-dimensional WAVE model, or for VECTIS calculations where a coupled solution with WAVE would provide more accurate boundary conditions. For further background and theory, consult the co-simulation manual available within the WAVE help manual.

In a WAVE-VECTIS coupled calculation, each VECTIS model is started as a separate child process by WAVE.

Current restrictions for the VECTIS model are as follows:

1. The domain containing coupling must be a single-phase fluid domain.
2. Participating boundaries must be defined as "Mass Flow".
3. All participating boundaries must belong to the same fluid domain.
4. The calculation must be unsteady.

It is recommended that the WAVE and VECTIS files be stored in separate directories. One convenient way to do this is to have "WAVE" and "VECTIS" subdirectories below the main project directory.

Typically a "run_VECTIS" script is used to execute VECTIS as a child process. An example script which changes to the VECTIS directory and starts "vsolve" could be as follows:

```
cd ../VECTIS
vsolve -np 2 project
```

The VECTIS input file requires the corresponding link (WAVE junction) references. These can be set via the "Boundary Regions" panel (See Fig (13.2)).

18 USER PROGRAMMING

18.1 Introduction And Overview

In general, a user may not be able to do everything that he wants with a CFD solver. For example the user may want to specify a particular boundary condition profile or modify the solver parameters directly. For this reason, a powerful set of *User Programmable Routines (UPR)* have been implemented into VECTIS-MAX solver to allows direct access to most of the solver data structure which in turn extends the capability of the VECTIS-MAX solver. This enables a user to write and compile their own code that runs with the solver as a shared object, enabling the user to modify or add new features to the solver.

The user may perform many tasks such as setting up boundary conditions, initial conditions, material properties or output results for a specific part of the domain. They can also be used to specify additional sources for the transport equations solved by VECTIS-MAX .

To use the UPR capabilities of VECTIS-MAX the user should have knowledge of CFD and knowledge of Fortran 95/2003 programming language which is used to write user programming code. It should be noted however that even though UPRs significantly enhance the VECTIS-MAX solver capability, it is not possible to deal with every CFD problem using UPRs.

User routines can be utilised at different stages during the simulation process. The five base user routines are described in section 18.2. These five routines act as wrappers for the user written code to interact with the solver.

The solver variables that can be accessed in the user programmable routines are described in section 18.3.

The set of *User Programmable Routines (UPR)* are explained in depth in section 18.4.

The methodology for writing a UPR, compiling and using a dynamically shared object is the subject of section 18.5.

Section 18.6 lists the messages that can be generated from the use of UPRs. These include errors, warnings and information messages.

Section 18.7 describes some examples of user programmable routines and their capability. The examples are written in the Fortran 95/2003 programming language.

18.2 Functionality And Calling Sequence

The UPRs enable access to the internal parameters of a simulation. The five different user programmable routines are:

1. *upr_generic()*
2. *upr_init()*
3. *upr_bnd_cond()*
4. *upr_properties()*
5. *upr_sources()*

The first UPR (upr_generic) is the generic UPR which is model independent. Most of the solver parameters can be accessed using this UPR. The other UPRs are more specific. These user programmable routines are called at different points in the simulation cycle. The purpose of each one is explained below.

1. *upr_generic()*: This is the generic user programming routine. It is called at five different stages of a simulation, mainly at the beginning of a simulation, before the start of each time step, after the end of each outer iteration, after the end of each time step and the end of the simulation. These five stages at which *upr_generic* may be utilised are given in Figure 18.1, labelled with 'upr_generic'. This generic UPR is used to implement general solver functions such as initialisation of variables (executed after the default initialisation), adjustment/manipulation of variables (called every time step or iteration), post-processing of results at the end of every time step/iteration, etc.

2. *upr_init()*: This UPR is used to specify initial cell values for VECTIS-MAX solution. It is called for each phase(species) present in fluid/solid material domains. Called once at the beginning of each simulation after reading the input file and before beginning the solution process. Any values set at this stage will overwrite values read from the input file.

3. *upr_bnd_cond()*: This user programmable routine is used to specify boundary conditions for the VECTIS-MAX solution variables. This is called from the same access point as *upr_init*. Boundary values set through the input file are overwritten at this stage of UPR. Other boundary profiles may also be defined. This UPR is called for each boundary region and each phase (species) present in the corresponding fluid/solid material domains.

4. *upr_properties()*: Thermo-physical properties of individual phase or species can be specified through the use of this user programmable routine. For example, if a property variable of a phase (species) can be modified then this UPR can be called to adjust the values of that variable. This UPR is called once at the start of simulation process and then it is called before the beginning of each solution iteration.

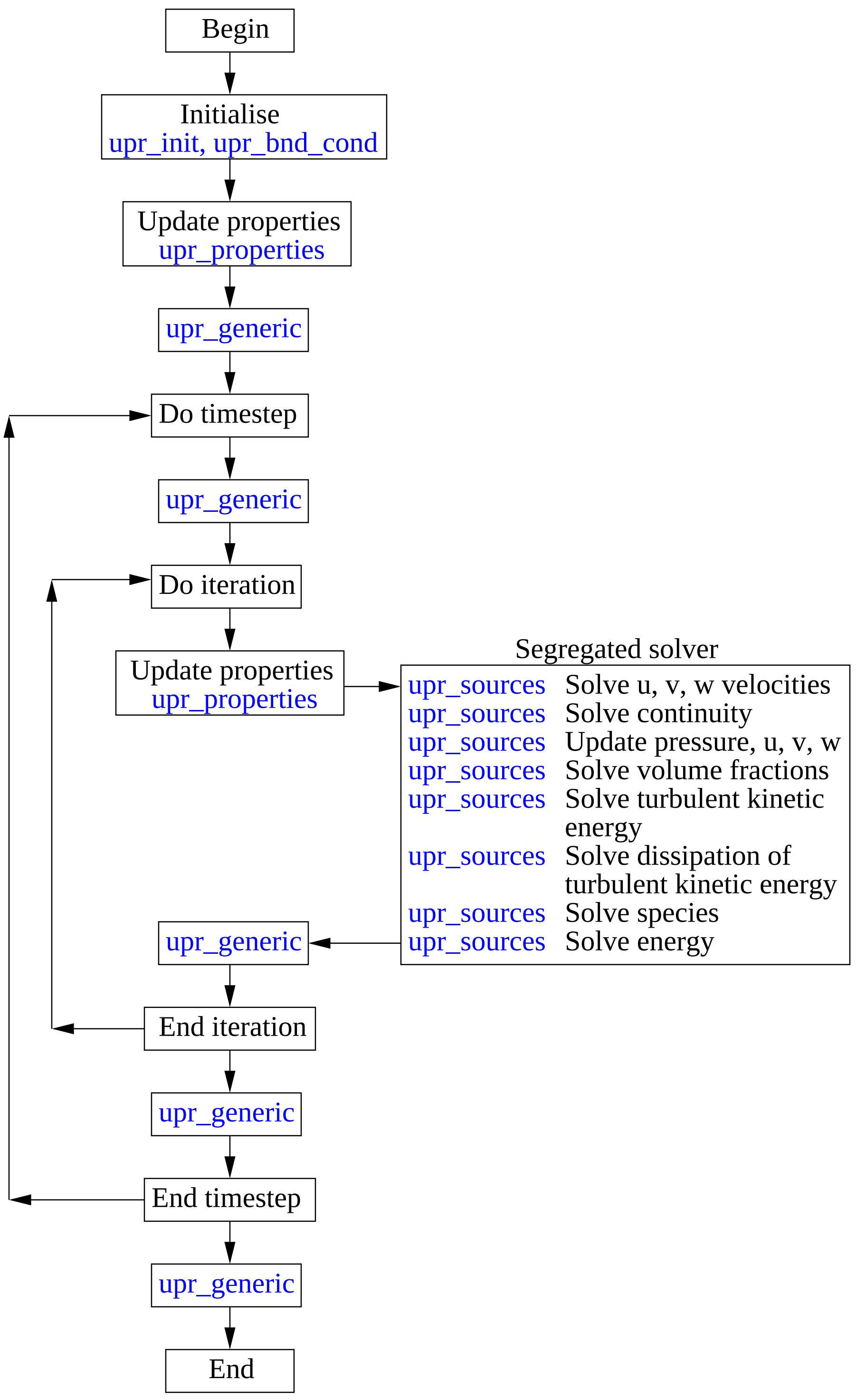

Begin
Initialise
upr_init, upr_bnd_cond
Update properties
upr_properties
upr_generic
Do timestep
upr_generic
Do iteration
Update properties
upr_properties
Segregated solver
upr_sources Solve u, v, w velocities
upr_sources Solve continuity
upr_sources Update pressure, u, v, w
upr_sources Solve volume fractions
upr_sources Solve turbulent kinetic energy
upr_sources Solve dissipation of turbulent kinetic energy
upr_sources Solve species
upr_sources Solve energy
upr_generic
End iteration
upr_generic
End timestep
upr_generic
End

Figure 18.1: Solver flow chart and the corresponding UPR stages (upr_ . . .).

5. *upr_sources()*: This UPR can be used to specify additional sources for each of the transport equations solved by VECTIS-MAX , except passive scalars which contain no source. This UPR is called for every solution outer iteration from within the SIMPLE segregated solver and it is called for each transport equation which is being solved by VECTIS-MAX .

18.3 Accessing Solver Variables

The following sections contain a brief description of available variables that a user can access. The user may access through UPRs *User Accessible Variables* (UAV) and *User Accessible Routines* (UAR).

18.3.1 User accessible variables

User accessible variables (UAV) are those variables that a user can access directly from the solver by specifying its identifier name. Most of these variables are used to identify certain property types of the VECTIS-MAX solver. For example, the UAV with the name *iget_cell* is used to identify cell values. Each of these identifiers is assigned an integer value to make the logic simpler. The user does not need to know its integer values. The purpose of each UAV will be much clearer through UPR examples provided. A complete and consistent list of all UAV is provided in this section in the form of tables. Each UAV is described in terms of its identifier name, description and the type (integer, real, character (cha)). A similar approach is used in UARs but in addition to the type of argument, a keyword 'pointer' is used where appropriate to indicate a pointer variable.

Precision format: Table 18.1 defines the numeric precision used in VECTIS-MAX . Both integer and real precision are defined. Both these precision format UAV can be used directly from UPRs.

Precision format		
Variable	Description	Type
iwp	integer working precision (32 bits)	integer, parameter
wph	real working double precision	integer, parameter

Table 18.1: Integer and real working precision format in VECTIS-MAX

Access groups: All user programming routines are classified into nacc_grp groups. Hence, each access group name in Table 18.2 identifies a UAR. These access group names are not used in UPRs but are mentioned here to illustrate the structure of user programming in VECTIS-MAX . For example *iacc_numbers* is associated with the UAR `get_number`.

Access groups		
Name	Description	Type
len_var_name	access variable name length (32)	integer, parameter
len_warn	length of warning message (256)	integer, parameter
iacc_number	main numbers	integer, parameter
iacc_domain	domain variables	integer, parameter
iacc_mat	material variables	integer, parameter
iacc_phase	phase variables	integer, parameter
iacc_spec	species variables	integer, parameter
iacc_ps	passive scalar variables	integer, parameter
iacc_reg	boundary & interface regions	integer, parameter
iacc_property	access properties	integer, parameter
iacc_grid_geom	grid geometry variables	integer, parameter
iacc_grid_connect	grid connectivity variables	integer, parameter
iacc_turb	turbulence constants	integer, parameter
iacc_run_ctrl	run control variables	integer, parameter
iacc_field	solution variable fields	integer, parameter
lacc_grp_name	lists names of access groups (currently there are 14 access groups)	character, (1:nacc_grp)

Table 18.2: Access groups

Various VECTIS-MAX objects: variable values can be obtained for various objects. A list of these objects available in VECTIS-MAX are given in Table 18.3.

Various VECTIS-MAX objects		
Variable	Description	Type
iget_cell	cell values	integer, parameter
iget_bnd	boundary face values	integer, parameter
iget_uif	upper material interface	integer, parameter
iget_lif	lower material interface	integer, parameter
iget_face	face values	integer, parameter
iget_o	cell old values	integer, parameter
iget_oo	cell old old values	integer, parameter
iget_dom	fluid/solid domain	integer, parameter
iget_mat	fluid solid material	integer, parameter
iget_phase	phase	integer, parameter
iget_specs	species	integer, parameter
iget_ps	passive scalars	integer, parameter
iget_breg	boundary region	integer, parameter
iget_ireg	interface region	integer, parameter
iget_part	partition (parallel run)	integer, parameter
iget_eq	transport equation	integer, parameter
iget_bctype	boundary condition type	integer, parameter
iget_mat_pro	material properties	integer, parameter
iget_ph_pro	fluid/solid phase properties	integer, parameter
iget_sp_pro	species properties	integer, parameter
iget_ps_pro	passive scalar properties	integer, parameter

Table 18.3: Various VECTIS-MAX objects

Post-processing variables: Table 18.4 provides the user with names of pre/post processing variables.

Post-processing variables		
Variable	Description	Type
i_cell	cells	integer, parameter
i_face	internal faces	integer, parameter
i_bnd	boundary faces	integer, parameter
i_intp	higher fluid/solid interface	integer, parameter
i_intm	lower fluid/solid interface	integer, parameter

Table 18.4: Names of post-processing variables

Other user accessible variables: Other parameters available directly to the user are given in Table 18.5. This lists for example different boundary condition types, global data (identifiers) related to transport equations and variables etc.

Some constants and numbers		
Variable	Description	Type
small	1.0E-30_wph	real, parameter
zero	0.0_wph	real, parameter
one	1.0_wph	real, parameter
Material id's		
Variable	Description	Type
len_mat_name	length of the material name = 24	integer, parameter
Argument pointer for array lbc()		
ibct	boundary condition type	integer, parameter
Types of boundaries		
ibinl	inlet boundary, given velocity vector	integer, parameter
ibout	outlet boundary	integer, parameter
ibsym	symmetry boundary	integer, parameter
ibwal	wall boundary (no-slip, slip)	integer, parameter
ibpre	pressure (static, total, average)	integer, parameter
ibtot	total (stagnation) boundary	integer, parameter
ibmas	inlet boundary, given mass flow rate	integer, parameter
Argument pointers for real array rbc()		
ibturin	turbulent intensity	integer, parameter
ibturls	turbulent length scale	integer, parameter
Global data (identifiers) related to transport equations (ieq)		
ifmom	id for fluid momentum equations	integer, parameter
ifmas	id for continuity (mass, pressure)	integer, parameter
iene	id for energy equation	integer, parameter
ifcs	id for concentration of species	integer, parameter
ifte	id for turbulent energy equation	integer, parameter
ifed	id for dissipation of turbulent energy	integer, parameter
ifvf	id for volume of fraction	integer, parameter
ifps	id for passive scalar calculation	integer, parameter
Property identifiers		
idens	density id	integer, parameter
ivis	viscosity id	integer, parameter

Table 18.5: Parameters accessible directly through UPRs.

Global data (identifiers) related to variables		
Variable	Description	Type
iu	id for fluid u-momentum	integer, parameter
iv	id for fluid v-momentum	integer, parameter
iw	id for fluid w-momentum	integer, parameter
ipr	id for fluid pressure	integer, parameter
ien	id for energy	integer, parameter
ics	id for species concentration	integer, parameter
ite	id for turbulent energy	integer, parameter
ied	id for turbulent dissipation rate	integer, parameter
ivit	id for turbulent vsicosity	integer, parameter
ips	id for passive scalar	integer, parameter

Table 18.6: Variable ids accessible directly through UPRs.

18.3.2 User accessible routines

In addition to the user accessible variables, a number of user accessible routines (UAR) are also available. These routines can be called from UPRs to exchange information with the solver kernel. Each UAR is listed in a table where the purpose of UAR and its arguments are explained. Thus a user only needs to look at the table to see which arguments need to be passed to the UAR and what result to expect. If the UAR can be called with optional arguments, all options are listed as subsections of the table. To distinguish types of arguments, input arguments are given in *italic* fonts, output arguments in **bold** and some tables have an input and output column defined explicitly. If a UPR is a function, its return type is also specified. Arguments may also be arrays in which case for example *array(:)* is a one dimensional array, *array(:,:)* a two dimensional array etc.

Here we give a number of examples to show how UARs can be called to exchange information with the solver. Note the the exclamation mark (!) indicates a comment in Fortran 95/2003.

Variables within quotation marks (" ") given in this section describe options available and should be defined as integers before use.

18.3.2.1 get_number()

This function is used to retrieve main variables and it is seen as a starting point in user programming. A number of variables related to the total number of domains, materials, phases, species, boundary & interface regions, geometric quantities such as number of cells, vertices etc. are given in Table 18.7. The following code:

```
integer(iwp) ::  n_doms
n_doms=get_number('n_domains')
```

declares `n_doms` to be of type integer and invokes the `get_number('n_domains')` function to return the number of domains in the simulation. A similar approach is used to get other main numbers defined in Table 18.7.

function **get_number**(*var_name*)	
Input	Output
var_name	**get_number**, return type: integer
n_domains	Total number of fluid and solid domains
n_fluid_doms	Number of fluid domains
n_solid_doms	Number of solid domains
n_mat_doms	Total number of materials
n_fluid_mats	Number of fluid materials
n_solid_mats	Number of solid materials
n_bnd_regs	Number of boundary regions
n_interf_regs	Number of interface regions
n_regions	Number of boundary and interface regions
n_phases	Total number of phases
n_fluid_phases	Number of fluid phases
n_solid_phases	Number of solid phases
n_species	Number of species
n_ps	Number of passive scalars
n_cells	Number of internal and halo (buffer) cells)
n_bnd_faces	Number of boundary faces
n_internal_face	Number of internal faces
n_mat_interface	Number of internal faces at material interfaces
n_vertices	Number of vertices
n_partitions	Number of partitions/processors
n_current_part	Current partition/processor number
n_halo_cells	Total number of halo (buffer) cells

Table 18.7: Function get_number(): to get values for main variables (numbers) defined in the access group 'iacc_numbers'

18.3.2.2 get_domain()

This UAR is used to retrieve information related to domains. Thus a list of information is returned for all domains. Information such as number of materials, indices of starting and ending phases, species, boundary & interface regions etc. (see Table 18.8 for a complete list) are provided for all domains. For example if a simulation case has 2 domains and domain 1 has 100 cells, domain 2 has 200 cells, then the following code:

```
integer(iwp),pointer ::  i1(:), i2(:)

call get\_domain('ise\_cell', i1,i2)
```

will give the following information:

```
i1(1)=1          i2(1)=100         for domain 1
i1(2)=101        i2(2)=300         for domain 2
```

Note that `i2(:)` is optional for some of the variables in Table 18.8 If it is used, its content will not be altered. Upper bounds for arrays `i1(:)`, `i2(:)` can be obtained using function get number().

`subroutine get_domain(` *var_name*, **i1**, **i2**)		
Input	Output	
var_name (cha)	**i1(:)** (integer, pointer,optional)	**i2(:)** (integer, pointer,optional)
n_dom_mat	number of materials for each domain, i1(1:n_domains)	-
list_dom_mat	list of domain materials, i1(1:2*n_mat_doms)	-
ise_phase	starting phase index, i1(1:n_domains)	ending phase index, i2(1:n_domains)
ise_bnd_reg	starting boundary region index, i1(1:n_domains)	ending boundary region index, i2(1:n_domains)
ise_specs	starting species index, i1(1:n_domains)	ending species index, i2(1:n_domains)
ise_ps	starting passive scalar index, i1(1:n_domains)	ending passive scalar index, i2(1:n_domains)
ise_cell	starting cell index, i1(1:n_domains)	ending cell index, i2(1:n_domains)
ise_bnd_face	starting boundary face index, i1(1:n_domains)	ending boundary face index, i2(1:n_domains)
ise_internal_face	stating internal face index, i1(1:n_domains)	ending internal face index, i2(1:n_domains)
ise_high_interface	starting higher interface index, i1(1:n_domains)	ending higher interface index, i2(1:n_domains)
ise_low_interface	starting lower interface index, i1(1:n_domains)	ending lower interface index, i2(1:n_domains)

Table 18.8: Subroutine get_domain(): to get values for variables defined in the access group 'iacc_domain' describing domain structure, mainly starting and ending indices of domain objects

18.3.2.3 get_mat()

This UAR is used to retrieve information related to materials. Information regarding material reference properties, starting and ending indices for different variables related to materials can be obtained though this UAR for all materials. From Table 18.9 it can be seen that this UAR can be called in 2 different ways (**a**-**b**).

a) The first way of calling only needs 2 arguments and the information obtained has to do with material reference properties. For example to get the reference pressure for all materials the following code can be used:

```
real(wph),pointer ::  p_ref(:)

call get\_mat('r_pressure',p_ref)
```

where p_ref(:) now contains all reference values for all materials.

b) The second way of calling needs 3 arguments where argument 2 and 3 need to be declared as integers and the information obtained is related to starting & ending indices. Consider a simulation with 2 materials where material 1 contains 2 fluid phases and material 2 contains 1 fluid phase. Indices of starting & ending phases over all materials can be obtained using the following:

```
integer(iwp),pointer ::  i1(:), i2(:)

call get_mat('ise_phase',i1,i2)
```

where i1(:) and i2(:) will have:

```
i1(1)=1          i2(1)=2          for material 1
i1(2)=3          i2(2)=3          for material 2
```

In a similar way to subroutine get_domain(), the upper bounds for arrays `rval(:)`, `i1(:)` and `i2(:)` can be obtained using function get number().

subroutine get_mat(*var_name*, **rval**)	
Input	Output
var_name (cha)	**rval(:)** (real, pointer)
r_temperature	reference temperature, rval(1:n_mat_doms)
r_pressure	reference pressure, rval(1:n_mat_doms)
r_gas_const	reference gas constant, rval(1:n_mat_doms)
r_area	reference area, rval(1:n_mat_doms)
r_length	reference length, rval(1:n_mat_doms)
r_velocity	reference velocity, rval(1:n_mat_doms)
r_density	reference density, rval(1:n_mat_doms)
r_viscosity	reference viscosity, rval(1:n_mat_doms)
r_spec_heat	reference specific heat, rval(1:n_mat_doms)
r_spec_heat_ratio	reference specific heat ratio, rval(1:n_mat_doms)

Table 18.9: Subroutine get_mat(): to get variables defined in the access group 'iacc_mat'; mainly reference material properties and starting and ending indices of material objects

`subroutine get_mat(` *var_name*, **i1**, **i2**)		
Input	Output	
var_name (cha)	**i1(:)** (integer, pointer,optional)	**i2(:)** (integer, pointer,optional)
mat_type	type: fluid or solid, i1(1:n_mat_doms)	-
mat_compress	compressibility flag, i1(1:n_mat_doms)	-
parent_dom	list of domains, i1(1:n_mat_doms)	-
ise_phase	starting phase, i1(1:n_mat_doms)	ending phase, i2(1:n_mat_doms)
ise_bnd_reg	starting boundary region i1(1:n_mat_doms)	ending boundary region i2(1:n_mat_doms)
ise_interf_reg	starting interface region, i1(1:n_mat_doms)	ending interface region, i2(1:n_mat_doms)
ise_specs	starting species, i1(1:n_mat_doms)	ending species, i2(1:n_mat_doms)
ise_ps	starting passive scalars, i1(1:n_mat_doms)	ending passive scalars, i2(1:n_mat_doms)
ise_cell	starting cell, i1(1:n_mat_doms)	ending cell, i2(1:n_mat_doms)
ise_bnd_face	starting boundary face , i1(1:n_mat_doms)	ending boundary face, i2(1:n_mat_doms)
ise_internal_face	starting internal face, i1(1:n_mat_doms)	ending internal face, i2(1:n_mat_doms)
ise_high_interface	starting high interface, i1(1:n_mat_doms)	ending high interface, i2(1:n_mat_doms)
n_low_interface	number of lower faces at material interfaces, i1(1:n_mat_doms)	-
isa_low_interface	initial address for lower interfaces in l_low_interface, i1(1:n_mat_doms)	-
l_low_interface	list (pointer) of faces at lower material interfaces, i1(1:n_interf_regs)	-
n_low_interf_reg	number of lower regions, i1(1:n_mat_doms)	-
isa_low_interf_reg	initial address for l_low_interf_reg, i1(1:n_mat_doms)	-
l_low_interf_reg	list of lower interface regions, i1(1:n_interf_regs)	-

Table 18.10: Subroutine get_mat(): continued from previous table

18.3.2.4 get_phase()

This UAR is used to retrieve information related to phases (see Table 18.11).

To establish the type of phases used then do the following:

```
integer(iwp),pointer ::  ph_type(:)
call get_phase('phase_type',ph_type)
```

Array `ph_type` may contain one of the following entries:

- □ "gas"=1
- □ "liquid"=2
- □ "solid"=3

If `ph_type(1)=1` indicates that phase 1 is a gas.

The compressibility options can be obtained as:

```
integer(iwp),pointer ::  ph_comp(:)
call get_phase('phase_compress',ph_comp)
```

The compressibility of each fluid/solid phase is defined by array `ph_comp(0:n_phases)`. The following parameters, identifying a fluid (solid) type in terms of compressibility, i.e. Mach number, can be assigned to `ph_comp(:)`:

- □ "incompres"=0– incompressible fluid phase or solid material: density is constant, compressibility coefficient $\mathscr{C} \to 0$.
- □ "iwcompres"=1– weakly compressible fluids: density may vary with temperature but the change of compressibility coefficients is relatively small. This definition includes liquids and gases, where the gases pressure changes relative to the reference pressure are small. In terms of the Mach number, weakly compressible fluid is defined for $Ma < 0.3$.
- □ "isubsonic"=2– subsonic gas flow: $0.3 \leq Ma \leq 1$.
- □ "isupsonic"=3– supersonic gas flow: $Ma > 1$.

The calculation options for mixture of species for fluid phases are obtained as:

```
integer(iwp),pointer ::  ph_mix_opts(:)
call get_phase('mixture_opts',ph_mix_opts)
```

Array `ph_mix_opts(:)` may have one of the following identifiers:

- □ "iphmix_none"=0– single-component phase

- □ “iphmix_vec“ =1– vectis calculation method
- □ “iphmix_wave“=2– combination of wave (density, Cp) and vectis for other properties
- □ “iphmix_wavec“=3– wave properties with combustion

The get the starting & ending species indices for all phases then:

```
integer(iwp),pointer ::  i1(:), i2(:)
call get_phase('ise_specs',i1,i2)
```

If for example phase 1 has 2 species and phase 2 has 3 species then arrays `i1(:)` and `i2(:)` will contain the following data:

```
i1(1) = 1        i2(1) = 2        for phase 1
i1(2) = 3        i2(2) = 5        for phase 2
```

`subroutine get_phase(` *var_name*, **i1**, **i2**)		
Input	Output	
var_name (cha)	**i1(:)** (integer, pointer,optional)	**i2(:)** (integer, pointer,optional)
phase_type	type (liquid,gas,solid), i1(1:n_phases)	-
phase_compress	compressibility for each phase, i1(1:n_phases)	-
mixture_opts	calculation options for mixture of species i1(1:n_phases)	-
ise_specs	starting species, i1(1:n_fluid_phases)	ending species, i2(1:n_fluid_phases)
ise_ps	starting passive scalar, i1(1:n_fluid_phases)	ending passive scalar, i2(1:n_fluid_phases)
ise_phase_bnd_reg	index of starting boundary region, i1(1:n_phases)	index of ending boundary region, i2(1:n_phases)
isa_phase_bnd_reg	starting allocation addresses for phase variables at boundary regions, i1(1:n_phases)	-
ise_phase_interf_-reg	index of starting interface region, i1(1:n_phases)	index of ending interface region, i2(1:n_phases)
isa_phase_interf_-reg	starting allocation addresses for phase variables at interface regions, i1(1:n_phases)	-

Table 18.11: Subroutine get_phase(): to get values for variables defined in the access group 'iacc_-phase', mainly start/end phase objects and phase properties

18.3.2.5 get_species()

This UAR is used to retrieve information related to species, mainly starting and ending boundary or interface regions (see Table 18.12).

Species are constituents of fluid phases. They are indexed respecting the order in which they appear in the ordered set of fluid phases. In order to store the region–wise species values, array addresses or 'species regions' are assigned to the first species and then subsequently for other species of the considered phase `iph`. By visiting all fluid materials `material=1,n_fluid_mats`, i.e. all fluid phases `iph=1,n_fluid_phases` there will be `nspreg` addresses. This number is the sum of species regions over all fluid domains, where the number of species regions within each domain is the product of the number of boundary regions and the number of domain species.

To obtain starting and ending addresses for boundary, region–wise values of species variables do the following:

```
integer(iwp),pointer ::  i1(:), i2(:)
call get_species('ise_specs_bnd_reg',i1,i2)
```

`subroutine get_species(` *var_name*, **i1**, **i2**)		
Input	Output	
var_name (cha)	**i1(:)** (integer, pointer,optional)	**i2(:)** (integer, pointer,optional)
ise_specs_bnd_reg	index of starting boundary region, i1(1:n_species)	index of ending boundary region, i2(1:n_species)
isa_specs_bnd_reg	starting allocation addresses for species at boundary regions, i1(1:n_species)	-
ise_specs_interf_-reg	index of starting interface region, i1(1:n_species)	index of ending interface region, i2(1:n_species)
isa_specs_interf_-reg	starting allocation addresses for species at interface regions, i1(1:n_species)	-

Table 18.12: Subroutine get_species(): to get values for variables defined in the access group 'iacc_-species', mainly start/end boundary/interface regions for species

18.3.2.6 get_ps()

This UAR is used to retrieve information related to passive scalars, mainly starting and ending boundary or interface regions (see Table 18.13).

Passive scalars are defined within fluid phases. They are indexed in a similar way to species.

To obtain starting and ending addresses for boundary, region–wise values of passive scalar variables do the following:

```
integer(iwp),pointer ::  i1(:), i2(:)
call get_ps('ise_ps_bnd_reg',i1,i2)
```

subroutine get_ps(*var_name*, **i1**, **i2**)		
Input	Output	
var_name (cha)	**i1(:)** (integer, pointer,optional)	**i2(:)** (integer, pointer,optional)
ise_ps_bnd_reg	index of starting boundary region, i1(1:n_ps)	index of ending boundary region, i2(1:n_ps)
isa_ps_bnd_reg	starting allocation addresses for species at boundary regions, i1(1:n_ps)	-
ise_ps_interf_reg	index of starting interface region, i1(1:n_ps)	index of ending interface region, i2(1:n_ps)
isa_ps_interf_reg	starting allocation addresses for species at interface regions, i1(1:n_ps)	-

Table 18.13: Subroutine get_ps(): to get values for variables defined in the access group 'iacc_ps', mainly start/end boundary/interface regions for passive scalars

18.3.2.7 get_reg()

This UAR is used to retrieve information related to boundary regions, species and passive scalar regions, boundary conditions as well as starting & ending indices of boundary faces for each boundary region. The call to this subroutine can be made in four different ways (**a** to **d**) as reflected by subsections of Table 18.14. Note that some variables used in this table and some other tables are for illustrative purpose only.

These variables/parameters are:

- □ nte: – number of transport equations (currently nte=14).

- ☐ nbcop: – number of integer control parameters to deal with boundary conditions (currently nbcop=8).
- ☐ nbcopr: – number of real boundary condition control parameters (currently nbcopr=9)
- ☐ nspreg: – number of species boundary regions.
- ☐ npsreg: – number of passive boundary scalar regions.
- ☐ nphreg: – number of phase boundary regions.
- ☐ nafve: – number of addresses for (internal) face vertices
- ☐ nabve: – number of addresses for boundary face vertices
- ☐ nproph: – number of phase properties (currently nproph=6)
- ☐ nprosp: – number of species properties (currently nprosp=11)
- ☐ nprops: – number of passive scalar properties (currently nprops=3)
- ☐ nturb_par: – number of turbulence control parameters (nturb_par=4)
- ☐ nbtype: – number of boundary types (currently nbtype=15)

Other variables given in plain text including those in Table 18.15 provide optional arguments with option value and need to be declared before using them.

a) To get the list of boundary regions then use the following:

```
integer(iwp),pointer ::  l_bc(:,:)
call get_reg('l_reg_cond',l_bc)
```

There are `nbcop` entries in the array `l_bc(:,:)` and for the given region index ’ir’ this array will return one of the following integer variables:

- ☐ “ibmat1“=0: – index of adjacent (parent in case of interfaces) material domain: `l_bc`(“ibmat1“,ir).
- ☐ “ibct“=1: – boundary condition type: `l_bc`(“ibct“,ir).
- ☐ “ibcop“=2: – boundary condition option related to the boundary type: `l_bc`(“ibcop“,ir).
- ☐ “ibreg“=3: – original boundary region index assigned in the pre–processor: `l_bc`(“ibreg“,ir)
- ☐ “ibinout“=4: – inflow or outflow condition: `l_bc`(“ibinout“,ir)=1 defines the outflow region, `l_bc`(“ibinout“,ir)=0 inflow and `l_bc`(“ibinout“,ir)$<$ 0 the wall or symmetry region.
- ☐ “ibcset“=5: – option to set–up boundary conditions: `l_bc`(“ibcset“,ir). Default uniform (i.e. region–wise) boundary conditions will be provided if `l_bc`(“ibcset“,ir)=0. Other options are: user–defined boundary conditions `l_bc`(“ibcset“,ir)=1, use of tabulated data `l_bc`(“ibcset“,ir)=2 and provision of time–dependent boundary conditions `l_bc`(“ibcset“,ir)=3.

- “ibcomp“=6: – region compressibility flag: The values assigned to the `l_bc`(“ibcomp“,ir) can indicate the incompressible `l_bc`(“ibcomp“,ir) = ”incompres”, weakly compressible `l_bc`(“ibcomp“,ir) = ”iwcompres”, Mach number $Ma \leq 0.3$), subsonic `l_bc`(“ibcomp“,ir) = ”isubsonic”, $0.3 < Ma < 1$) and supersonic flow condition `l_bc`(“ibcomp“,ir) = ”isupsonic”, $Ma \geq 1$).
- “ibmat2“=7: – index of neighbouring material domain in case of interface regions. Note that `l_bc`(“ibmat1“,ir) < `l_bc`(“ibmat2“,ir).

During the solution procedure the initial boundary values can be fixed or can be updated, depending on the boundary type. The data retrieved using `l_reg_opts` option:

```
integer(iwp), pointer ::  r_opts(:,:)
call get_reg('l_reg_opts',r_opts)
```

determines what kind of update (if any) will be done. Thus, for every equation `ieq=1` to `nte` and region `ir=1` to `n_regions`, the following integer parameter can be assigned to `r_opts(ieq,ir)`:

- “ibmirr“=0: – perform zero–order extrapolation (mirror values)
- “ibzerog“=1: – enforce the zero–gradient boundary condition
- “ibextr“=2: – perform the 2nd order extrapolation from inside
- “ibflux“=3: – calculate the boundary variable from its flux
- “ibfix“=4: – keep values fixed and , respectively
- “ibexpr“=5: – use another expression

b) To get a list of region boundary values for species then use:

```
real(wph), pointer ::  sp_val(:)
call get_reg('specs_value',sp_val)
```

Array sp_val is used in conjunction with species indices in Table 18.12 (see subroutine get_species) as:

```
integer ::  ir   ! region index
integer ::  is   ! species index
real    ::  bval
integer(iwp),pointer ::  is_sp_bndreg(:), ie_sp_bndreg(:)
integer(iwp),pointer ::  a_sp_bndreg(:)

 !get index of starting & ending boundary region for species
call get_species('ise_specs_bnd_reg',is_sp_bndreg,ie_sp_bndreg)
 !get the starting allocation addresses for species at
 !boundary regions
call get_species('isa_specs_bnd_reg',a_sp_bndreg)

do ir=is_sp_bndreg(is), ie_sp_bndreg(is)
   bval = sp_val(ir+a_sp_bndreg(is))
end do
```

Now bval represents the boundary value at region ir for species with index is. Data for option specs_flux are used in a similar way to specs_value. For passive scalar option: ps_-value and ps_flux, index pointer from Table 18.13 are used.

Additional boundary region data (real type variables) are provided by addresses in the array l_-reg_value(0:nbcopr,0:n_regions). For the considered region ir and each address, the array l_reg_value(:,:) stores the following region–wise data:

- □ “ibflo“=1: – mass flow rate: l_reg_value(“ibflo“,ir).
- □ “ibmachv=2: – boundary region Mach number: l_reg_value(“ibmach“,ir).
- □ “ibptot“=3: – total pressure: l_reg_value(“ibptot“,ir).
- □ “ibttot“=4: – total (stagnation) temperature: l_reg_value(“ibttot“,ir).
- □ “ibrwh“=5: – wall roughness height: l_reg_value(“ibrwh“,ir).
- □ “ibrwc“=6: – turbulence constant for rough walls: l_reg_value(“ibrwc“,ir).
- □ “ibsplit“=7: – flow splitting factor at the outlets: l_reg_value(“ibsplit“,ir).
- □ “ibturin“=8: – turbulence intensity: l_reg_value(“ibturin“,ir).
- □ “ibturls“=9: – turbulence length scale: l_reg_value(“ibturls“,ir).

c) To get the list of region boundary conditions then do the following:

```
real(wph),pointer ::  rval(:,:)
call get_reg('l_reg_value',rval)
```

To get region boundary values for phases then do the following:

```
real(wph),pointer ::  ph_val(:,:)
call get_reg('phase_value',ph_val)
```

Index pointers in Table 18.11 are needed for array ph_val (see subroutine get_phase) and is used as follows:

```
integer(iwp),pointer ::  is_ph_bndreg(:), ie_ph_bndreg(:)
integer(iwp),pointer ::  a_ph_bndreg(:)

 !get index of starting & ending boundary region for phases
call get_phase('ise_phase_bnd_reg',is_ph_bndreg,ie_ph_bndreg)
 !get the starting allocation addresses for phase variables
 !at boundary regions
call get_phase('isa_phase_bnd_reg',a_ph_bndreg)

do ir=is_ph_bndreg(iph), ie_ph_bndreg(iph)
   ph_val(ivar, ir+a_ph_bndreg(iph))
end do
```

where `ivar` is the variable identifier, `ir` is the region index and `iph` is the phase index.

d) To get starting and ending boundary faces for each boundary region then:

```
integer(iwp),pointer ::  i1(:), i2(:)
call get_reg('ise_reg_face',i1, i2)
```

`subroutine get_reg(` *var_name*, **ival**)		
Input	Output	
var_name (cha)	**ival(:,:)** (integer, pointer)	
l_reg_cond *l_reg_opts*	boundary region control parameters, ival(1:nbcop,1:n_regions) boundary value option, ival(1:nte,1:n_regions)	
`subroutine get_reg(` *var_name*, **rval**)		
Input	Output	
var_name (cha)	**rval(:)** (real, pointer)	
specs_value *specs_flux* *ps_value* *ps_flux*	region boundary values for species, rval(1:nspreg) region boundary fluxes for species, rval(1:nspreg) region boundary values for passive scalars, rval(1:npsreg) region boundary fluxes for passive scalars, rval(1:npsreg)	
`subroutine get_reg(` *var_name*, **rval**)		
Input	Output	
var_name (cha)	**rval(:,:)** (real, pointer)	
l_reg_value *vel_direction* *reg_flux* *phase_value*	region boundary condition array, rval(1:nbcopr,1:n_regions) region boundary velocity direction, rval(1:3,1:n_regions) region boundary fluxes, rval(1:nte,1:n_regions) region boundary values for phases, rval(1:nte,1:nphreg)	
`subroutine get_reg(` *var_name*, **i1**, **i2**)		
Input	Output	
var_name (cha)	**i1(:)** (integer, pointer)	**i2(:)** (integer, pointer)
ise_reg_face	starting boundary face for each boundary region, i1(1:n_regions)	ending boundary face for each boundary region, i2(1:n_regions)

Table 18.14: Subroutine get_reg(): to get integer variables defined in the access group 'iacc_region'.

18.3.2.8 get_property()

This UAR is used to retrieve information related to phase, species and passive scalar material properties and reference values. Table 18.16 is split in two sections and explained here (**a**-**b**).

a) To get the list of phase properties calculation options then do the following:

```
integer(iwp),pointer ::  ph_pro_opts(:,:)
call get_property('l_phase_opts',ph_pro_opts)
```

Array `ph_pro_opts(1:nproph,1:n_phases)` now contains all calculation option for phase properties. For example ph_pro_opts(1,2) represents a density option for phase 2. The next list specifies parameters used to define fluid properties which are required when solving transport equations of either individual phases or the mixture of phases:

- ☐ *idens=1*: – density. Calculation options: “ipro_const“, “ipro_mix“, “ipro_igas“, “ipro_poly“, “ipro_user“, “ipro_bouss“.
- ☐ *ivis=2*: – laminar viscosity. Calculation options: “ipro_const“, “ipro_mix“, “ipro_poly“, “ipro_power“, “ipro_suth“, “ipro_user“, “ipro_invis“, “ipro_nonwt“.
- ☐ “icon“=3: – thermal conductivity (if the energy equation is solved). Calculation options: “ipro_const“, “ipro_mix“, “ipro_poly“, “ipro_suth“, “ipro_user“.
- ☐ “icph“=4: – temperature–averaged specific heat either at constant pressure (the total enthalpy equation is solved) or constant volume (the total energy equation solved for). Calculation options: “ipro_const“, “ipro_mix“, “ipro_igas“, “ipro_poly“, “ipro_user“.
- ☐ “imolw“=5: – molecular weight, which is used to calculate the gas constant. Calculation options: “ipro_const“, “ipro_mix“, “ipro_user“.
- ☐ “itexc“=6: – volumetric thermal expansion coefficient, used to calculate the density for small temperature ranges. Calculation options: “ipro_const“, “ipro_user“.

In case of the multicomponent fluid phase, i.e. when species mass fraction equations are solved, additional fluid (species) properties have to be defined:

- ☐ “idifm“=7: – mass diffusion coefficient, which governs the laminar mass diffusion flux. Calculation options: “ipro_const“, “ipro_poly“, “ipro_user“.
- ☐ “idift“=8: – thermal diffusion coefficient, which governs the mass diffusion flux caused by the temperature gradient (Soret effect). Calculation options: “ipro_const“, “ipro_poly“, “ipro_user“.
- ☐ “iheatf“=9: – formation (standard state or heat of formation) enthalpy for reacting flow problems involving solution of the energy equation. Calculation options: “ipro_const“, “ipro_user“.
- ☐ “itempf“=10: – formation (standard–state reference) temperature for reacting flow problems involving solution of the energy equation. Calculation options: “ipro_const“, “ipro_user“.
- ☐ “itsch“=11: – turbulent Schmidt number, which is used to calculate turbulent mass diffusion flux. Calculation options: “ipro_const“, “ipro_user“.

To get the list of species properties calculation options then do the following:

```
integer(iwp),pointer ::  species_pro_opts(:,:)
call get_property('l_species_opts',species_pro_opts)
```

The solution of the energy equation for solid materials requires the same properties as for fluid phases: the density, thermal conductivity and specific heat. The calculation options for these properties are restricted to the constant or user–defined values.

In summary, considering solution of phase equations (including solid phases) nproph=6 properties may be needed: density, viscosity, thermal conductivity, specific heat, molecular weight, thermal expansion coefficient, id's from *idens=1* to "itexc"=6. Multicomponent fluid phases involve nprosp=11 physical properties: density, viscosity, thermal conductivity, specific heat, molecular weight, thermal expansion coefficient, mass diffusion coefficient, thermal diffusion coefficient, formation enthalpy, formation reference temperature and turbulent Schmidt number (id's from *idens=1* to "itexc"=6, and from "idifm"=7 to "itsch"=11).

For passive scalar (nprops=3) properties are defined:

- "idens_ps"=1: – density. Calculation options: "ipro_const", "ipro_mix", "ipro_igas", "ipro_poly", "ipro_user", "ipro_bouss".
- "idifm_ps"=2: – mass diffusion coefficient, which governs the laminar mass diffusion flux. Calculation options: "ipro_const", "ipro_poly", "ipro_user".
- "itsch_ps"=3: – turbulent Schmidt number, which is used to calculate turbulent mass diffusion flux for passive scalar Calculation options: "ipro_const", "ipro_user".

To get the list of passive scalar properties calculation options then do the following:

```
integer(iwp),pointer ::  passca_pro_opts(:,:)
call get_property('l_ps_opts',passca_pro_opts)
```

According to the thermodynamic state of matter, the properties of the fluid component can be a function of temperature and pressure. The current solver version has built-in options to deal with thermally perfect fluids, i.e. the physical properties can be a function of temperature only (not of pressure). The exception is the density of an ideal gas. The user–defined properties can be a function of both temperature and pressure.

The arrays `ph_pro_opts(0:nproph,0:n_phases)`, `species_pro_opts(0:nprosp,0:n_species)` and `passca_pro_opts(0:nprops,0:n_ps)` control evaluation of the phase, species and passive scalar properties, respectively. For example, the density `ipro=idens` of a phase `iph` will be calculated according to the option flag returned by `ph_pro_opts(ipro,iph)`. If `ph_pro_opts(idens,iph)= 6`("ipro_suth") then density is calculated according to Sutherland law.

A number of control parameters have been introduced in order to cover a wide range of calculation options which can be assigned to the arrays `ph_pro_opts(0:nproph,0:n_phases)`,

Calculation options for fluid/solid properties	
Property type	Description
”ipro_inact”=0	inactive property
”ipro_const”=1	constant properties
”ipro_mix” =2	mixture of fluid species
”ipro_igas” =3	ideal gas or ideal gas mixture
”ipro_poly” =4	temperature dependent, polynomial
”ipro_power”=5	temperature dependent, power law
”ipro_suth” =6	temperature dependent, Sutherland law
”ipro_user” =7	supplied by user subroutine
”ipro_expr” =8	expression supplied by user
”ipro_invis”=9	inviscid fluid (for viscosity)
”ipro_nonwt”=10	non Newtonian fluid
”ipro_bouss”=11	Boussinesq approx for density
”ipro_isgas”=12	isentropic gas
”ipro_wave” =13	mixture properties from WAVE
”ipro_expo” =14	temperature dependent, exponential

Table 18.15: Calculation options for variables describing fluid/solid properties

`species_pro_opts(0:nprosp,0:n_species)` and `passca_pro_opts(0:nprops,0:n_-ps)`. The physical property can be inactive (“ipro_inact“=0), constant (“ipro_const“=1) or solution dependent, i.e. a function of temperature and/or pressure. Note that properties of the multi-component phase are usually solution dependent, even for the constant properties of constituent species. For such properties, the “ipro_mix“=2 calculation option should be used, and phase properties will be calculated from Equation (8.39). However, if it is desirable to specify the composition independent property (not a function of mass fractions) of the multicomponent fluid, this property will have the calculation option different from “ipro_mix“.

The ideal gas model, identified by “ipro_igas“=3 option, can be used in many practical situations to closely approximate properties of compressible gas flow. Typically, it is used to calculate the density and specific heat.

The temperature dependence for any property can be expressed by a general polynomial function, option “ipro_poly“=4, see Equation (8.26). For some properties, such as the molecular viscosity, it is sometimes more appropriate to use the power–law form, “ipro_power“=5, see Equation (8.4). Additional option available for the calculation of viscosity and thermal conductivity in terms of temperature is *Sutherland* formula for gases, “ipro_suth“=6, see Equation (8.3).

The next two options, labelled as the user–defined property “ipro_user“=7 and user–supplied expression “ipro_expr“=8 are applicable to any property. The last option is not implemented yet. The options “ipro_invis“=9 and “ipro_nonwt“=10 are related to the treatment of molecular viscosity for an inviscid and non–Newtonian fluid, respectively. The option “ipro_nonwt“ invokes calculation of generalised non-Newtonian viscosity and it is not yet implemented.

The last calculation option, “ipro_bouss“=11, enables approximation of density by Boussinesq’s formula, Equation (8.27). To apply this formula, the constant density at reference temperature ($\rho_{ref}(T_{ref})$) and the corresponding thermal expansion coefficient β are required.

It is important to realise that an option is applicable to the certain properties only, see the specification of properties above.

b) The constant or reference property values of phases, species and passive scalars can be obtained as (see Table 18.16):

```
real(wph),pointer ::  ph_pro_ref(:,:)
real(wph),pointer ::  sp_pro_ref(:,:)
real(wph),pointer ::  ps_pro_ref(:,:)

call get_property('r_phase_values',ph_pro_ref)
call get_property('r_species_values',sp_pro_ref)
call get_property('r_ps_values',ps_pro_ref)
```

For example `ph_pro_ref(ivis,iph)` represents the reference value of viscosity for phase iph.

`subroutine get_property(` *var_name*, **ival**)	
Input	Output
var_name (cha)	**ival(:,:)** (integer, pointer)
l_phase_opts	property calculation options for phases, ival(1:nproph,1:n_phases)
l_species_opts	property calculation options for species, ival(1:nprosp,1:n_species)
l_ps_opts	property calculation options for passive scalars, ival(1:nprops,1:n_ps)
`subroutine get_property(` *var_name*, **rval**)	
Input	Output
var_name (cha)	**rval(:,:)** (real, pointer)
r_phase_values	reference values for phase properties, rval(1:nproph,1:n_phases)
r_species_values	reference values for species properties, rval(1:nprosp,1:n_species)
r_ps_values	reference values for passive scalar properties, rval(1:nprops,1:n_ps)

Table 18.16: Subroutine get_property(): to get values for variables defined in the access group 'iacc_-pro', mainly phase/species/passive scalars calculation options and reference properties

18.3.2.9 get_grid_geom()

This UAR is used to retrieve information related to grid objects. There are two section in Table 18.17 and are explained here (**a**-**b**).

a) To get the cell volume for all CV then:

```
real(wph),pointer ::  cell_volumes(:)
call get_grid_geom('cell_vol',cell_volumes)
```

Array `cell_volumes(1:n_cells)` contains the cell volumes for all cells `n_cells`. To get the upper bound `n_cells` and other upper bounds in Table 18.17 then use `get_number` function. Alternatively, use Fortran 95/2003 utilities such as `ubound(array,dim)` and `lbound(array,dim)` for upper and lower array bounds: `max_cells = ubound(cell_-volumes,1)`

where `max_cells = n_cells`.

b) To get the x,y,z co-ordinates of vertices then do the following:

```
real(wph),pointer ::  vertices(:,:)
call get_grid_geom('xyz_vert',vertices)
```

Starting and ending interface indices are needed for each material domain to be able to use `interf_dist_v` and `interf_normal_d`. Example:

```
integer(iwp),pointer ::   i1(:), i2(:)
real(wph),pointer ::      i_dist_vec(:,:)

call get_mat('ise_high_interface', i1,i2)
call get_grid_geom('interf_dist_v',i_dist_vec)
```

Now `i_dist_vec(1:3, i1(1):i2(n_mat_doms))` represents all distance vector from a cell to interface for all material domains. If distance vector from a cell to interface for material domain 1 are required then use `i_dist_vec(1:2, i1(1):i2(1))`.

A similar approach is used for option `interf_normal_d`.

18.3.2.10 get_grid_connect()

This UAR is used to retrieve information related to grid connectivity. This UPR can be called in 2 different ways as represented in Table 18.18 and explained here (**a**-**b**).

a) Consider the example given below:

```
integer(iwp),pointer ::   f_verts(:)
integer(iwp),pointer ::   internal_face_vert(:)
integer(iwp),pointer ::   isa_fv(:)

call get_grid_connect('n_face_verts',f_verts)
call get_grid_connect('l_face_verts',internal_face_verts)
call get_grid_connect('isa_face_verts',isa_fv)
```

Array `internal_face_verts` now contains a list of face vertices for all internal faces. For the face `j` with `k=1,f_verts(j)` vertices, the particular vertex is given as `kv=internal_-face_verts(isa_fv(j)+k)` where array `isa_fv` provides the starting address in the array `internal_face_verts(:)`, which provides the list of face vertices.

To get a list of vertices for each boundary face then:

`subroutine get_grid_geom(` *var_name*, **rval**)	
Input	Output
var_name (cha)	**rval(:)** (real, pointer)
cell_vol	cell volume, rval(1:n_cells)
bndf_normal_d	normal distance from the near-boundary cell centre to the boundary face, rval(1:n_bnd_faces)
face_weight	weighting (interpolation) factor, rval(1:n_internal_face)

`subroutine get_grid_geom(` *var_name*, **rval**)	
Input	Output
var_name (cha)	**rval(:,:)** (real, pointer)
xyz_vert	x,y,z coordinates of vertices, rval(1:3,1:n_vertices)
xyz_face_c	position vector at cell face centre, rval(1:3,1:n_internal_face)
xyz_bndf_c	position vector at boundary face centre, rval(1:3,1:n_bnd_faces)
xyz_cell_c	cell centre coordinates, rval(1:3,1:n_cells)
face_surf_v	cell-face surface vector, rval(1:3,1:n_internal_face)
bndf_surf_v	boundary face surface vector, rval(1:3,1:n_bnd_faces)
face_dist_v	distance vector from cell 1 to 2, rval(1:3,1:n_internal_face)
bndf_dist_v	distance vector from a cell to boundary face, rval(1:3,1:n_bnd_faces)
interf_dist_v	distance vector from a cell to interface, rval(1:3,ise_high_interface[i(1):i2(n_mat_doms)])
interf_normal_d	normal distance from the near-wall cell centre to the interface between fluid and solid, rval(1:2,ise_high_interface[i(1):i2(n_mat_doms)])

Table 18.17: Subroutine get_grid_geom(): to get grid geometry variables defined in access group 'iacc_grid_geom'

```
integer(iwp),pointer ::   b_verts(:)
integer(iwp),pointer ::   bnd_face_vert(:)
integer(iwp),pointer ::   isa_bv(:)

call get_grid_connect('n_bndf_verts',b_verts)
call get_grid_connect('l_bndf_verts',bnd_face_verts)
call get_grid_connect('isa_bnd_verts',isa_bv)
```

For the face `jb` with `k=1,b_verts(jb)` vertices, the particular vertex is returned as `kv=bnd_face_verts(isa_bv(jb)+k)`, where array `isa_bv` provides starting address to be used in the array `lbver(:)`, which provides the list of boundary face vertices.

```
integer(iwp),pointer ::   nc_faces(:,:)
integer(iwp),pointer ::   cell_faces(:,:)
integer(iwp),pointer ::   isa_cf(:,:)

call get_grid_connect('n_cell_faces',nc_faces)
call get_grid_connect('l_cell_faces',cell_faces)
call get_grid_connect('isa_cell_faces',isa_cf)
```

b) To get the number of faces that enclose each cell then:

```
integer(iwp),pointer ::   nc_faces(:,:)
call get_grid_connect('n_cell_faces',nc_faces)
```

For the given cell `ic`, `nc_faces(1,ic)` returns a number of faces whose normal vectors point out of the cell (upper cell faces) while `nc_faces(2,ic)` gives a number of faces with normals pointing into the cell (lower cell faces).

To get a list of faces that enclose each cell then:

```
integer(iwp),pointer ::   cell_faces(:,:)
integer(iwp),pointer ::   isa_cf(:,:)

call get_grid_connect('l_cell_faces',cell_faces)
call get_grid_connect('isa_cell_faces',isa_cf)
```

For the cell `ic` with `jc=1,nc_faces(1,ic)` upper faces, their indices are obtained as `j=cell_faces(1,isa_cf(1,ic)+jc)`. Similarly, lower cell faces are given as `j=cell_faces(2,isa_cf(2,ic)+jc)`, where `jc=1,nc_faces(2,ic)`. Arrays `isa_cf(1,ic)` and `isa_cf(2,ic)` return starting addresses for upper and lower faces in array `cell_faces` , respectively.

`subroutine get_grid_connect(` *var_name*, (integer, pointer)**ival**)	
Input	Output
var_name (cha)	**ival(:)** (integer, pointer)
l_bndf_cells	list of cells adjacent to a boundary face, ival(1:n_bnd_faces)
n_face_verts	number of face vertices for each face, ival(1:n_internal_face)
isa_face_verts	initial (starting) address for face vertices, ival(1:n_internal_face)
l_face_verts	list of face vertices for all internal faces, ival(1:nafve)
n_bndf_verts	number of boundary face vertices, ival(1:n_bnd_faces)
isa_bndf_verts	initial address for boundary face vertices, ival(1:n_bnd_faces)
l_bndf_verts	list of vertices for a boundary face, ival(1:nabve)
`subroutine get_grid_connect(` *var_name*, **ival**)	
Input	Output
var_name (cha)	**ival(:,:)** (integer, pointer)
l_face_cells	list of 2 cells adjacent to an internal face, ival(1:2,1:n_internal_face)
n_cell_faces	number of faces which enclose a cell, ival(1:2,1:n_cells)
isa_cell_faces	starting address for cell faces, ival(1:2,1:n_cells)
l_cell_faces	list of faces that enclose a cell, ival(1:2,1:n_internal_face)

Table 18.18: Subroutine get_grid_connect(): to get grid connectivity variables defined in access group 'iacc_grid_connect'

18.3.2.11 get_turb()

This UAR is used to retrieve information related to turbulence model identifiers and constants. Table 18.19 is split in 3 sections and represented as **a-c** here.

a) Turbulence modelling control parameters can be obtained as:

```
integer(iwp),pointer ::   turb_par(:,:)
call get_turb('turb_ctrl_vars',turb_par)
```

Array `turb_par(1:nturb_par,1:n_domains)` now contains a list of turbulence control variables. Number of turbulence control parameters is (`nturb_par=4`) and are identified by pointers as:

- ”iturb_meth”=1: – physical method: DNS, RANS, LES, DES. Physical model identified by ”iturb_meth” returns one of the following:
  - ”iturb_inv”=-1: – inviscid flow, no turbulence models
  - ”iturb_none”=0: – no modelling,laminar/DNS
  - ”iturb_rans”=1: – Reynolds Averaged Navier-Stokes models
  - ”iturb_les”=2: – Large Eddy Simulations
  - ”iturb_des”=3: – Detached Eddy Simulations

  For example if `turb_par`(”iturb_meth”,2)=0 then the physical model is laminar/DNS for domain 2.

- ”iturb_modf”=2: – model family belonging to a method. Family model identified by ”iturb_-modf” returns one of the following:
  - ”iinvisc” =-1: – inviscid flow
  - ”idns” =0: – dns/laminar, no modelling
  - ”ievm_zeroeq”=1: – zero-equation (algebraic) models
  - ”ievm_oneeq” =2: – one-equation models
  - ”ievm_keps” =3: – k-epsilon models
  - ”ievm_kom” =4: – k-omega models
  - ”ievm_v2f” =5: – Durbin’s keps v2-f model
  - ”iarsm” =6: – algebraic Reynolds stress models
  - ”idrsm” =7: – Differential Reynolds stress models
  - ”iles” =8: – Large-Eddy Simulation models
  - ”ides” =9: – Hybrid RANS/LES, (DES)

  For example if `turb_par`(”iturb_modf”,2)=3 then the family model for domain 2 is the k-epsilon model.

- □ ”iwall_meth”=3: – near-wall method. Two options are available:
  - ”iwallf”=1: – wall modelling with wall functions
  - ”ilowre”=2: – low-Reynolds number modelling

  For example if `turb_par`(”iwall_meth”,2)=1 then near-wall method used for domain 2 is wall modelling with wall functions

- □ ”iwall_modl”=4: – near-wall model from a given method. Three options are available:
  - ”iwallf_std”=1: – standard wall functions
  - ”iwallf_sca”=2: – scalable wall functions
  - ”iwallf_uwb”=3: – unified wall boundary conditions

  For example if `turb_par`(”iwall_modl”,2)=2 then the wall function type used for domain 2 is the scalable wall function.

b) To get C_μ or any other turbulence constant then do the following:

```
real(wph),pointer ::   c_mu(:)
call get_turb('cmu',c_mu)
```

Array `c_mu(:)` now contains C_μ values for each domain.

c) Note the use of a 2D array in `eps_const` and `prandtl_number` in Table 18.19.

18.3.2.12 get_run_ctrl()

This UAR is used to retrieve information related to run control variables. This UAR can be called in 2 ways represented with 2 sections in Table 18.20.

a) The following example:

```
integer(iwp) ::   curr_iter
call get_run_ctrl('current_iter',curr_iter, 1)
```

returns the current iteration `curr_iter`.

b) The get the name of the current project that is running then:

```
character(len=*) ::   proj_name
call get_run_ctrl('project_name',proj_name)
```

Variable `proj_name` now contains the name of the project that is running.

`subroutine get_turb(` *var_name*, **ival**)	
Input	Output
var_name (cha)	**ival(:,:)** (integer, pointer)
turb_ctrl_vars	list of turbulence control variables, ival(1:nturb_par,1:n_domains)
`subroutine get_turb(` *var_name*, **rval**)	
Input	Output
var_name (cha)	**rval(:)** (real, pointer)
cmu *cappa* *log_law_const* *yptrans* *ypvisc*	C_mu, rval(1:n_fluid_doms) von Karman const, rval(1:n_fluid_doms) log_law constant, rval(1:n_fluid_doms) nondimensional wall distance, rval(1:n_fluid_doms) viscous sub-layer non-dim thickness, rval(1:n_fluid_doms)
`subroutine get_turb(` *var_name*, **rval**)	
Input	Output
var_name (cha)	**rval(:,:)** (real, pointer)
eps_const *prandtl_number*	k-eps, eps-eq const, rval(1:n_domains,1:6) turb prandtl numbers rval(1:n_domains,1:nte)

Table 18.19: Subroutine get_turb(): to get turbulence variables defined in access group 'iacc_turb'

`subroutine get_run_ctrl(` *var_name*, **ival**, *id*)		
Input		Output
var_name (cha)	*id* (integer)	**ival** (integer)
current_iter *end_iter* *current_tstep* *end_tstep* *post_freq*	id for the global domain	current iteration, ival ending iteration, ival current time step, ival ending time step, ival frequency for printing into project files, ival
`subroutine get_run_ctrl(` *var_name*, **proj_nums** (character))		
Input	Output	
var_name (cha)	**proj_nums**	
project_name *proj_run_number*	name of current project, proj_nums current project run number, proj_nums	

Table 18.20: Subroutine get_run_ctrl(): to get 'run control' variables defined in access group 'iacc_run_ctrl'

18.3.2.13 get_name()

This UAR is used to retrieve the solver object name (see Table 18.21).

The list of variables under *iget* in Table 18.21 are predefined in VECTIS-MAX and are readily available.

To get the name of domain 1 (idt=1) then do the following:

```
character(len=*) ::   dom_name
call get_name(idt,dom_name,iget_dom)
```

Variable `dom_name` contains the name of the domain.

To get the name of species 2 (idt=2) then do the following:

```
character(len=*) ::   specs_name
call get_name(idt,specs_name,iget_specs)
```

To get the name of an equation with idt=4 then

```
character(len=*) ::   eqn_name
call get_name(idt,eqn_name,iget_eq)
```

Variable `eqn_name` may contain one of the following transport equation names (idt given in brackets for illustration): *momentum* (ifmom), *mass_pressure* (ifmas), *energy* (iene), *spec_mass_-fracs* (ifcs), *turb_energy* (ifte), *dissipation* (ifed), *volume_fracs* (ifvf) and *passive_scalar* (ifps).

Phase, species and passive scalar property names that can be obtained with option *iget_ph_pro*, *iget_sp_pro* and *iget_ps_pro* were explained in get_property subroutine.

Phase and species list the following: density, lam_viscosity, conductivity, specific_heat, gas_-constant, thermal_exp_coeff, lam_mass_diffusion, thermal_diff_coeff, enthalpy_form_heat, en-thalpy_form_temp, turb_schmidt_numb where phase can have the first 6 property names.

In case of passive scalars then the following list is obtained: density, lam_mass_diffusion and turb_schmidt_numb

18.3.2.14 get_name_list()

This UAR can be used to obtain a list of names for each VECTIS-MAX object (see Table 18.22).

The list of material names is obtained as:

```
character(len=*),pointer ::  mat_names
call get_name_list(mat_names,iget_mat)
```

where character array `mat_names(:)` now contains a list of all material names. A similar way is used for all other *iget* options in Table 18.22.

`subroutine get_name(`*idt*, **cname**, *iget*)		
Input		Output
idt range (integer)	*iget* (integer)	**cname** (cha)
1:n_domains	*iget_dom*	domain name
1:n_mat_doms id	*iget_mat*	material name
-n_phases:-1	*iget_phase*	phase name
1:n_species	*iget_specs*	species name
1:n_ps	*iget_ps*	passive scalar name
1:n_bnd_regs	*iget_breg*	boundary region name
1:n_interf_regs	*iget_ireg*	interface region name
1:nte	*iget_eq*	equation name
1:nbtype	*iget_bctype*	name of boundary condition type
1:nproph	*iget_ph_pro*	name of phase property
1:nprosp	*iget_sp_pro*	name of species property
1:nprops	*iget_ps_pro*	name of passive scalar property

Table 18.21: Subroutine get_name(): to get VECTIS-MAX object name

`subroutine get_name_list(` **lname**, *iget*)	
Input	Output
iget (integer)	**lname(:)** (cha, pointer)
iget_dom	list of domain names
iget_mat	list of material names
iget_phase	list of phase names
iget_specs	list of species names
iget_ps	list of passive scalar names
iget_breg	list of boundary regions
iget_ireg	list of interface regions

Table 18.22: Subroutine get_name_list(): Get a list of names lname(:) for each VECTIS-MAX object

18.3.2.15 get_id()

This UAR is used to retrieve the solver object id (see Table 18.23).

In subroutine get_name we showed how to obtain a solver object name by providing its id. This section deals with the case whereby we specify the solver object name to obtain the solver object id.

To obtain the id of a phase with name 'water' do the following:

```
character(len=*) ::   ph_name
call get_id(ph_name,idt,iget_phase)
```

Now idt contains the id of phase with name 'water'.

In order to make use of options: *iget_eq*, *iget_ph_pro*, *iget_sp_pro* and *iget_ps_pro* refer to subroutine get_name for relevant names.

subroutine get_id(*cname*, **idt**, *iget*)		
Input		Output
cname (cha)	*iget* (integer)	**idt** (integer)
domain name	*iget_dom*	domain id
material name	*iget_mat*	material id
phase name	*iget_phase*	phase id
species name	*iget_specs*	species id
passive scalar name	*iget_ps*	passive scalar id
boundary region name	*iget_breg*	boundary region id
interface region name	*iget_ireg*	interface region id
equation name	*iget_eq*	equation id
boundary condition type name	*iget_bctype*	boundary condition type id
phase property name	*iget_ph_pro*	phase property id
species property name	*iget_sp_pro*	species property id
passive scalar property name	*iget_ps_pro*	passive scalar property id

Table 18.23: Subroutine get_id(): to get VECTIS-MAX objects id

18.3.2.16 get_parent()

This UAR can be used to obtain parent identifier and optionally a name for the vectis object *idt* described by the identifier *iget* (see Table 18.24).

To obtain the parent id of a phase with *idt* do the following:

```
integer(iwp)      ::   idt, parent_id
character(len=*)  ::   parent_name

idt=1  ! phase id
call get_parent(idt,iget_phase,parent_id, parent_name)
```

Note that `parent_name` is optional.

A similar approach is used for other options of *iget*.

18.3.2.17 eq_idt()

This UAR can be used to return identifier for derived data types related to the given equation *ieq* and for the object *iobj* (domain, phase, etc.) specified by the *iget* (see Table 18.25).

subroutine get_parent(*idt*, *iget*, **par_id**, **par_name**)			
Input		Output	
idt (integer)	*iget* (integer)	**par_id** (integer)	**par_name** (cha, optional)
domain id	*iget_dom*	domain id	
material id	*iget_mat*	material id	
phase id	*iget_phase*	phase id	parent name
species id	*iget_specs*	species id	
passive scalar id	*iget_ps*	passive scalar id	
boundary region id	*iget_breg*	boundary region id	
interface region id	*iget_ireg*	interface region id	

Table 18.24: Subroutine get_parent(): to get parent identifier

If the domain type does not exist null value is returned, `eq_idt=0`. For example, if you want to access momentum equation variables for fluid material domain `mat=2`, the command:

```
idt=eq_idt(ieq=ifmom,iobj=mat,iget=iget_mat)
```

will return *idt* which is then used to access members of momentum type, eg. the velocity field is described by: `tmom(idt)%u`. Note, however, that not all variables, i.e. members of derived type, are used in the given run. For example, the old velocity vector field within the momentum type does not exist for a steady run. Since all type members are Fortran 95/2003 pointers you can check if the variable values are available by using the Fortran 95/2003 logical inquiry function `associated(pointer)`. Thus, `associated(tmom(idt)%u)` returns either `.true.` or `.false.` value.

Furthermore *iobj* is related to *iget*, i.e. *iget_dom* is used then *iobj* must be domain *id* etc.

function **eq_idt**(*ieq,iobj,iget*)		
Arguments		Description
eq_idt	⇒	(integer) index of domain type for the given equation ieq
iobj	⇒	(integer) either domain or material or phase or species or passive scalar index
ieq	⇒	(integer) equation id
iget	⇒	(integer) describes object type; takes in: *iget_dom*, *iget_mat*, *iget_phase*, *iget_specs*, *iget_ps*

Table 18.25: Function eq_idt(): to return identifier *idt* for derived data types

18.3.2.18 get_field()

This UAR is used to get field values either for a scalar or a vector for domain type (*idt*) (see Table 18.27). If *idt* is not known then use `eq_idt` UAR to get *idt*. A complete list of names that can be passed to `get_field` subroutine through *var_name* are given in Table 18.26 together with availability of variables at grid objects.

There are 2 different options for `get_field` as specified in 2 sections in Table 18.27.

a) The first method is used to obtain field values at one grid object.

Example: Obtain velocity values at cell centre, firstly obtain domain type idt for momentum and then obtain velocity values:

```
real(wph), pointer ::  vel_cell(:,:) ! cell velocity values
integer            ::  idt, dom      ! 'dom' is domain id

dom = 1      ! domain 1 used as an example

 !Get velocity domain index
idt=eq_idt(ifmom,dom,iget_dom)

 !get cell velocity values
call get_field(idt,iget_cell,'velocity',vel_cell)
```

Note the use of a 2D array for velocity `vel_cell(:,:)`. For a scalar a 1D array must be used.

Argument *iget_uif* corresponds to upper interface values. These are *fi* values at cell faces along material interfaces whose normal vectors point out of the given material domain. Similarly *iget_lif* corresponds to lower interface values. These are *fi* values at cell faces along material interfaces whose normal vectors point into the given material domain.

b) The second method is used to obtain field values at one or more grid object. Thus a number of optional arguments are available in the second part of Table 18.27.

Example: Obtain pressure field at cell centre, boundary faces, lower interface and upper interface; firstly obtain domain type idt for pressure and then obtain pressure values:

```
integer :: idt, dom      ! 'dom' is domain id
integer :: scal_field(1) !rank of variable (here it is a scalar)

real(wph),pointer  :: p(:)   &!cell pressure
                     ,pb(:)  &!boundary pressure
                     ,pi(:)  &!upper interface pressure
                     ,pli(:)  !lower interface pressure

scal_field  = (/1/)  !scalar rank
dom = 1              !domain 1 used as an example

 !Get pressure domain index
idt=eq_idt(ifmas,idom,iget_dom)

 !get pressure values
call get_field(idt,scal_field,'pressure' &
               ,fi_c=p,fi_b=pb,fi_ui=pi,fi_li=pli)
```

It is important to note that the starting & ending indices of field values can be obtained using get_domain subroutine. In the above example for velocity field, the indices of starting & ending cell for all domains can be obtained as:

```
integer(iwp),pointer ::  ic_s(:), ic_e(:)
call get_domain('ise_cell', ic_s, ic_e)
```

To loop over cells for domain `dom=1` then use:.

```
real ::  u, v, w

do i=ic_s(dom), ic_e(dom)
   u = vel_cell(1,i)  !x velocity component
   v = vel_cell(2,i)  !y velocity component
   w = vel_cell(3,i)  !z velocity component
end do
```

Indices can also be passed when the call to `get_field` is made as:

```
integer  :: ic1, ic2

ic1 = ic_s(dom)
ic2 = ic_e(dom)
call get_field(idt,iget_cell,'velocity',vel_cell(1:3,ic1:ic2))
```

One can also use Fortran 95/2003 `ubound()/lbound()` explained earlier.

18.3.2.19 get_grad()

This UAR can be used to obtain the gradient of scalar or vector solution variables. The user is repsonsible for allocating memory for the gradient array.

The first part of the Table 18.28 deals with scalar gradient and the second part with vector gradient.

a) Obtaining a scalar gradient: To get the gradient of temperature then do the following:

```
integer           :: dom, idt, n_cells
real(wph),pointer :: g_temperature(:,:)

dom = 1                         !domain 1 used as an example

!Get temperature domain index
idt=eq_idt(ifmas,idom,iget_dom)

!Allocate gradient array
n_cells = get_number('n_cells')
allocate(g_temperature(3,n_cells))

 !get gradient of temperature
call get_grad(idom,idt,g_temperature,'temperature')
```

The call to `get_grad` can also be made as:

```
call get_grad(idom,idt,gfi_out=g_temperature &
              ,var_name='temperature')
```

where pointers are assigned directly. Note that optional arguments that correspond to *fi_c*, *fi_b*, *fi_-ui* and *fi_li* need to be declared as pointers before using them, similar to `g_temperature(:,:)` used in the example above:

Momentum variables								
var_name	Meaning	Availability						
		cell	bnd	uif	lif	face	o	oo
velocity	velocity	x	x	x	x		x	x
density	density	x	x	x	x		x	x
eff_viscosity	effective viscosity $(\mu_l + \mu_t)$	x	x	x	x			
lam_viscosity	laminar viscosity	x						
mass_flow_rate	mass flow rate		x			x		
gas_constant	gas constant	x						
drho_dp	derivative of density over pressure (constant temperature)	x						
Pressure variables								
pressure	pressure	x	x	x	x		x	x
tot_pressure	total pressure		x					
tot_temperature	total temperature		x					
sat_pressure	saturation pressure	x						
Turbulence variables								
turb_energy	turbulent kinetic energy	x	x	x	x		x	x
dissipation	dissipation rate of turbulent kinetic energy	x	x	x	x		x	x
Energy variables								
temperature	temperature	x	x	x	x		x	x
tot_enth_energy	total enthalpy or total energy	x	x	x	x		x	x
lam_conductivity	laminar conductivity	x	x					
specific_heat	specific heat	x	x					
heat_flux	heat flux		x	x				
nuc_heat_flux	boiling heat flux		x	x				
sat_temperature	saturation temperature	x						
latent_heat	latent heat of evaporation	x						
surface_tension	surface tension	x						
Volume fraction variables								
phase_vol_frac	phase volume fraction	x	x	x	x		x	x
vof_mass_flow	volume fraction mass flow rate		x			x		
Species variables								
spec_mass_frac	species mass fraction	x	x	x	x		x	x
lam_mass_diff	laminar mass diffusion of species	x	x					
Passive scalar variables								
mass_frac_ps	mass fraction of passive scalar	x	x	x	x		x	x
lam_mass_diff_ps	laminar mass diffusion of passive scalar	x	x					

Table 18.26: List of accessible variables
This list provides names or keywords that may be used to get values for transport equations being solved and other variables that are derived from solved transport equations. Keywords given under *var_name* can be used with `function get_field` to obtain their values.

`subroutine get_field`(*idt*, *iget*, *var_name*, **fi_out**)			
Arguments			Description
idt	⇒		(integer) domain type id
iget	⇒		(integer) takes in: *iget_cell*, *iget_o*, *iget_oo*, *iget_bnd*, *iget_uif*, *iget_lif*, *iget_face*
var_name	⇒		(character) variable name (token)
fi_out()	⇒		(real, pointer) can either be a scalar of vector; depending on *iget*, the output would be: *fi* at centre, *fi* at centre (old), *fi* at centre (old old), *fi* at boundary, *fi* at upper interface, *fi* at lower interface and *fi* at faces respectively.
	fi_out(:)	→	output is a scalar
	fi_out(:,:)	→	output is a vector; similar to fi(:) but now the result is a vector (used for velocity for example)

`subroutine get_field`(*idt,var_rank,var_name*, **fi_c,fi_b,fi_ui,fi_li,fi_c_o,fi_c_oo,fi_f,var_id**)			
Arguments			Description
var_rank()	⇒		(integer) rank of the variable array
	var_rank(:)	→	if *var_name* is a scalar, rank is one
	var_rank(:,:)	→	if *var_name* is a vector, rank is two
fi_c()	⇒		(real, pointer, optional) cell variable associated with *var_name*
	fi_c(:)	→	if *var_name* is a scalar
	fi_c(:,:)	→	if *var_name* is a vector
fi_b()	⇒		(real, pointer, optional) boundary variable
	fi_b(:)	→	if *var_name* is a scalar
	fi_b(:,:)	→	if *var_name* is a vector
fi_ui()	⇒		(real, pointer, optional) variable at upper material interface
	fi_ui(:)	→	if *var_name* is a scalar
	fi_ui(:,:)	→	if *var_name* is a vector
fi_li()	⇒		(real, pointer, optional) variable at lower mat interface
	fi_li(:)	→	if *var_name* is a scalar
	fi_li(:,:)	→	if *var_name* is a vector
fi_c_o()	⇒		(real, pointer, optional) cell variable, old
	fi_c_o(:)	→	if *var_name* is a scalar
	fi_c_o(:,:)	→	if *var_name* is a vector
fi_c_oo()	⇒		(real, pointer, optional) cell variable, old-old
	fi_c_oo(:)	→	if *var_name* is a scalar
	fi_c_oo(:,:)	→	if *var_name* is a vector
fi_f()	⇒		(real, pointer, optional) variable at internal face
	fi_f(:)	→	if *var_name* is a scalar
	fi_f(:,:)	→	if *var_name* is a vector
var_id	⇒		(integer, optional) variable index

Table 18.27: Subroutine get_field(): to get field values either for a scalar or a vector

```
real(wph),pointer  :: g_temperature(:,:) &
                      t(:)    &!cell temperature
                     ,tb(:)   &!boundary temperature
                     ,ti(:)   &!upper interface temperature
                     ,tli(:)   !lower interface temperature

 !get gradient of temperature
call get_grad(idom,idt,gfi_out=g_temperature &
              ,fi_c=t,fi_b=tb,fi_ui=ti,fi_li=tli)
```

b) Obtaining a vector gradient: To get the gradient of velocity components then:

```
integer           ::  dom, idt
real(wph),pointer ::  gx_vel(:,:) &!gradient of x component
                     ,gy_vel(:,:) &!gradient of y component
                     ,gz_vel(:,:)  !gradient of z component

dom = 1     !domain 1 used as an example

 !Get velocity domain index
idt=eq_idt(ifmom,idom,iget_dom)

 !get gradient of velocity
call get_grad(idom,idt,gx_vel,gy_vel,gz_vel,var_name='velocity')
```

It is important to note the following information related to *var_name*:

1. if *var_name* is supplied then there is no need to pass optional arguments such as: *fi_c*, *fi_b*, *fi_ui* and *fi_li*. The routine will automatically find the values of these optional arguments.
2. if *var_name* is not supplied and for example we want the gradient of a user defined variable then *fi_c* should be supplied at least.
3. if arguments *fi_b*, *fi_ui* and *fi_li* are not passed then the routine will use 'near boundary cell values' as 'boundary values'.

18.3.2.20 deln_star()

This UAR can be used to calculate non-dimensional wall distance based on the wall function approach (see Table 18.29). Refer to Equation (9.26) for mathematical description.

Consider an example where boundary region with id `ir=1`, fluid material `mat=1` and `idt=1`. The following example illustrates the use of `deln_star` function.

subroutine get_grad(*idom*, *idt*, **gfi_out**, *var_name*, *fi_c*, *fi_b*, *fi_ui*, *fi_li*)		
Arguments		Description
idom	⇒	(integer), fluid/solid domain id
idt	⇒	(integer), domain type index
gfi_out(:,:)	⇒	(real, pointer), scalar gradient of *var_name*
var_name	⇒	(character, optional), variable name
fi_c(:)	⇒	(real, optional) scalar variable cell values
fi_b(:)	⇒	(real, optional) scalar boundary values
fi_ui(:)	⇒	(real, optional) scalar upper-interface values
fi_li(:)	⇒	(real, optional) scalar lower-interface values
subroutine get_grad(*idom*, *idt*, **gx_out**, **gy_out**, **gz_out**, *var_name*, **fi_c**, **fi_b**, **fi_ui**, **fi_li**)		
Arguments		Description
gx_out(:,:)	⇒	(real, pointer) grad vector x-component
gy_out(:,:)	⇒	(real, pointer) grad vector y-component
gz_out(:,:)	⇒	(real, pointer) grad vector z-component
fi_c(:,:)	⇒	(real, pointer, optional) vector variable cell values
fi_b(:,:)	⇒	(real, pointer, optional) vector boundary values
fi_ui(:,:)	⇒	(real, pointer, optional) vector upper-interface values
fi_li(:,:)	⇒	(real, pointer, optional) vector lower-interface values

Table 18.28: Subroutine get_grad(): to get gradient of scalar or vector variables

```
integer ::  ir, j, ic, mat, idt
integer(iwp),pointer ::  jsbc(:), jsbc(:), l_bcells(:)
real(wph),pointer ::  dnb(:)
real(wph) :: ystar   !non-dimensional wall distance

 !get starting & ending boundary face for each boundary region
call get_reg('ise_reg_face',jsbc, jsbc)

 !get list of cells adjacent to a boundary face
call get_grid_connect('l_bndf_cells',l_bcells)

 !get of normal distance from the near-boundary
 !cell centre to the boundary face
call get_grid_geom('bndf_normal_d',dnb)

 !Calculate non-dimensional wall distance for all boundary
 !faces belonging to boundary region ir
do j=jsbc(ir),jebc(ir)
   ic=l_bcells(j)
   ystar=deln_star(idt,mat,ic,j,dnb(j))
   . . .
end do
```

function `deln_star`(*idt*,*mat*,*ic*,*j*,*deln*)		
Arguments		Description
deln_star	⇒	(real) normalised wall distance
idt	⇒	(integer) data type (domain or phase)
mat	⇒	(integer) fluid material id
ic	⇒	(integer) near-boundary cell (*l_bndf_cells*)
j	⇒	(integer) boundary face or interface (*ise_reg_face*)
deln	⇒	(real) normal distance cell->boundary (*bndf_normal_d* and *interf_normal_d*)

Table 18.29: Function deln_star(): to calculate non-dimensional wall distance.

18.3.2.21 local_force()

This UAR can be used to calculate to calculate viscous and pressure forces at individual boundary faces and interfaces (see Table 18.30)

Consider an example where boundary region with id `ir=1`. The following example illustrates the use of `local_force` subroutine.

```
integer ::  ir, j, ic, idt, dom
integer(iwp),pointer ::  jsbc(:), jsbc(:)
integer            :: scal_field(1) !scalar rank
integer            :: vect_field(1,1)!vector rank
real(wph),pointer :: pb(:)         !boundary pressure
real(wph),pointer :: vis_b(:)   !boundary effective viscosity
real(wph),pointer :: vol_frac_b(:) !boundary volume fraction
real(wph),pointer :: vel_cell(:,:) &!cell velocity
                     ,vel_bnd(:,:) &!boundary velocity
                     ,vel_ui(:,:)  &!upper interface velocity
                     ,vel_li(:,:)   !lower interface velocity

real(wph),pointer :: bfs_vec(:,:) !boundary face surface vector
real(wph),pointer :: dnb(:)

real(wph))         :: fpre(1:3)   &!pressure force
                    ,fvis(1:3)   &!viscous force
                    ,are          !boundary face area

scal_field  = (/1/)      !scalar rank
vect_field  = reshape((/1,1/),(/1,1/))  !vector rank
dom = 1      !domain 1 used as an example

 !Get pressure domain index
idt=eq_idt(ifmas,idom,iget_dom)
 !Get boundary pressure field
call get_field(idt,iget=iget_bnd,var_name='pressure',fi_out=pb)

 !Get velocity domain index
idt=eq_idt(ifmom,dom,iget_dom)
 !Get velocity at cell, boundary, lower and upper interface
call get_field(idt,vec_field,var_name='velocity',fi_c=vel_cell &
               ,fi_b=vel_bnd,fi_ui=vel_ui,fi_li=vel_li)

 !Get boundary effective viscosity field
call get_field(idt,iget=iget_bnd,var_name='eff_viscosity' &
               ,fi_out=vis_b)

 !Get phase domain index
idt=eq_idt(ifvf,idom,iget_phase)
 !Get boundary volume fraction
call get_field(idt,iget=iget_bnd,var_name='phase_vol_frac' &
               ,fi_out=vol_frac_b)

 !get boundary face surface vector
call get_grid_geom(var_name='bndf_surf_v', bfs_vec)

 !get of normal distance from the near-boundary
 !cell centre to the boundary face
call get_grid_geom('bndf_normal_d',dnb)

 !get starting & ending boundary face for each boundary region
call get_reg('ise_reg_face',jsbc, jsbc)

 !Calculate local pressure and viscous forces for all boundary
 !faces belonging to boundary region ir
do j=jsbc(ir),jebc(ir)
  ic=l_bcells(j)

  call local_force(j,ic,pb(j),vel_cell(:,ic),vel_bnd(:,j) &
                   ,vis_b(j), vof_frac_b(j),bfs_vec(:,j) &
                   ,dnb(j),fpre,fvis,area)
   . . .
end do
```

`subroutine local_force`(*j*,*ic*,*pj*,*uc*,*uj*,*vitj*,*vofj*, *sj*,*deln*,**fpre**,**fvis**,**area**)

Arguments		Description
j	⇒	(integer) boundary face or interface
ic	⇒	(integer) near-boundary cell
pj	⇒	(real) boundary pressure
uc(1:3)	⇒	(real) cell velocity
uj(1:3)	⇒	(real) boundary/interface velocity
vitj	⇒	(real) boundary effective viscosity
vofj	⇒	(real) boundary volume fraction
sj(1:3)	⇒	(real) boundary/interface surface vector
deln	⇒	(real) normal distance cell->face
fpre(1:3)	⇒	(real) pressure force
fvis(1:3)	⇒	(real) viscous force
area	⇒	(real) boundary face area

Table 18.30: Subroutine local_force(): to calculate viscous and pressure forces at individual boundary faces & interfaces

18.3.2.22 user_post()

This UAR can be used to register user post-processing data to stored at the user-selected post-processing write frequency (see Table 18.31). This can be done initially, e.g. within `upr_init` or within an `if-block` if using `upr_generic`.

Consider the following example:

```
subroutine upr_generic(id,icall_pos)
integer(iwp),intent(in) :: id        &!global domain id (==1)
                          ,icall_pos !calling position with solver
real(wp), pointer, save :: upr_data(:) !Pointer to user data
integer(iwp)            :: idt         !Domain type id
integer(iwp)            :: d_type      !Data type (e.g. cell based)
character(len=*)        :: d_name      !Data base name
integer(iwp),pointer    :: ic_s(:) &!Cell start id's for each domain
                          ,ic_e(:)  !Cell end id's for each domain
integer(iwp)            :: ic1, ic2

if (icall_pos == icp_beg_run) then
  !Get domain cell indices
  get_domain('ise_cell', ic_s, ic_e)

  !Allocate upr_data for domain 1 - and leave allocated
  d_type = i_cell
  d_name = 'test_data'
  idt    = 1
  ic1    = ic_s(idt)
  ic2    = ic_e(idt)
  allocate(upr_data(ic1:ic2))

  !register user data for domain idt
  call user_post(uprdata=upr_data,idt,d_type,d_name)
end if

if (icall_pos == icp_end_iter) then
  !Set upr_data to something...
  upr_data = ?
end if
end subroutine upr_generic
```

User data contained in array `upr_data(:)` is saved to the post-processing file. Data type `d_type` is used to establish whether data written is cell-based `(d_type=i_cell)`, boundary-based etc. (see Table 18.4 for further details).

The data in the post file is identified under the data base name `d_name`. Domain type id is used for output levels. If data base name is `USER_PHASE` and data is written for 2 phases then USER_PHASE is identified under `USER_PHASE_1` corresponding to phase 1 and `USER_PHASE_2` corresponding to phase 2 in the post file. Note that `upr_data(:)` is a 1D array. It can also be used as a 2D array depending on the variable.

`subroutine user_post`(*uprdata,idt,dtype,dname*)		
Arguments		Description
uprdata()	$\Rightarrow$	(real, pointer) store pointer of the user data
	uprdata(:) $\rightarrow$	(real) rank one array
	uprdata(:,:) $\rightarrow$	(real) rank two array
idt	$\Rightarrow$	(integer) store domain type id
dtype	$\Rightarrow$	(integer) store data type (e.g. cell based)
dname	$\Rightarrow$	(character) store data base name

Table 18.31: Subroutine user_post(): to store user data

18.3.2.23 global_sum()

This UAR can be used to get global sum over partitions used in parallel runs.

Table 18.32 is split into 2 sections. Each section has different arguments.

The first section deal with scalar variables whereas the second section deal with vector variables and vector variables can have 1 or 2 dimensions.

a) Get global sum for scalars: Suppose that we want to get the region area calculated over all partitions. Then using section 1 of the table, `global_sum(`**`svalue1,svalue2`**`)` we get:

```
real(wph) :: sum_area   !total region area

 !calculate 'sum_area' for each partition

 !get global sum over partitions
call global_sum(sum_area)
```

Variable `sum_area` now contains the total region area. Note that svalue2 is optional and is intended for use with 2 scalars, each of them returning its global sum over partitions.

b) Get global sum for vectors: To calculate mean region pressure use section 2 of the table, `global_sum(`**`values`**`,nvalues1,nvalues2)` as:

```
real(wp)  ::  p_reg(:)  !(mean) region pressure
integer   ::  nreg

 !get number of regions
nreg = get_number(n_bnd_regs)

 !calculate 'p_reg()' for each partition
 . . .

 !Get sums over all partitions
call global_sum(p_reg,nreg)
```

Here `p_reg(1:nreg)` passed to `global_sum` is a 1D array. A 2D array can also be used when required.

The examples presented above use a `real` type argument for `sum_area` and `p_reg()` but `integer` and `double precision` are also available.

18.3.2.24 global_max(), global_min()

This UAR can be used to obtain maximum/minimum over partitions. Table 18.33 is split into 2 sections. Each section has `global_max` and `global_min` and has support for `character`, `integer`, `real` and `double precision` format.

The first section deal with scalar variables whereas the second section deal with vector variables.

a) Get global maximum/minimum for scalars:

subroutine global_sum(***svalue1, svalue2***)			
Arguments			Description
svalue1	⇒		(integer) global sum
svalue2	⇒		(integer, optional) global sum
svalue1	⇒		(real) global sum
svalue2	⇒		(real, optional) global sum
svalue1	⇒		(double precision) global sum
svalue2	⇒		(double precision, optional) global sum
subroutine global_sum(***values***, *nvalues1, nvalues2*)			
Arguments			Description
values()	⇒		global sum: **values()** can either be 1D or 2D:
	values(:)	→	(integer) sum
	values(:)	→	(real) sum
	values(:)	→	(double precision) sum
	values(:,:)	→	(integer) sum
	values(:,:)	→	(real) sum
	values(:,:)	→	(double precision) sum
nvalues1	⇒		(integer) number of elements in dimension 1, **values**(1:nvalues1)
nvalues2	⇒		(integer, optional) number of elements in dimension 2, **values**(1:nvalues1,1:nvalues2). Used if **values()** is a 2D array.

Table 18.32: Subroutine global_sum(): to get global sum over partitions

```
real(wph) ::  mach_max  &!estimated maximum Mach number

 !calculate 'mach_max' for each partition
 . . .

 !Get global maximum value over all partitions
call global_max(mach_max)
```

Global minimum `global_min` is used in a similar way.

b) Get global maximum/minimum for vectors:

```
real(wp),pointer ::  vec(:)  !vector variable
integer          ::  n       !total entries in vec(1:n)

 !calculate 'vec()' for each partition
 . . .

 !Get global maximum values over all partitions
call global_max(vec, n)
```

`subroutine global_max(`***svalue***`)`, `subroutine global_min(`***svalue***`)`		
Arguments		Description
svalue	⇒	global max/min
	svalue →	(character) max/min
	svalue →	(integer) max/min
	svalue →	(real) max/min
	svalue →	(double precision) max/min
`subroutine global_max(`***values***,*nvalues*`)`, `subroutine global_min(`***values***,*nvalues*`)`		
Arguments		Description
values()	⇒	global max/min
	values(1:nvalues) →	(character) max/min
	values(1:nvalues) →	(integer) max/min
	values(1:nvalues) →	(real) max/min
	values(1:nvalues) →	(double precision) max/min
nvalues	⇒	(integer) number of elements for array **values()**

Table 18.33: Subroutine global_max(), Subroutine global_min(): to get max/min over partitions

18.3.2.25 concat_array()

This UAR can be used to take local array from each partition and concatenate into global array on partition=1, only order by partition number (see Table 18.34). Output can be optional global array or local array on partition 1.

The first part of the table deals with the case when a 1D array is used, whereas the second part if a 2D array is used. Each part of the table deals with 3 data types: integer, real and double.

The following example illustrates the use of a real array `t_val` in `concat_array` subroutine:

```
    !number of stored values on each partition
integer(iwp),allocatable,save :: dsize(:)
    !total number of stored values
integer(iwp),save :: gsize
    !compressed array of temperature
real(wph),allocatable,save ::  t_val(:)

integer(iwp) :: i_part&  !this partitions number
                n_part&  !number of paritions
                n        !counter for compressed arrays

n_part=get_number('n_partitions')
i_part=get_number('n_current_part')

allocate(dsize(n_part))
dsize=0
dsize(i_part)=n !number per partition

 !get global sum over partitions
call global_sum(dsize,n_part)

gsize=sum(dsize(:)) !total number

 !allocate memory for compressed array of temperature,
 !on partition 1 globally sized arrays are created.
if(i_part==1)then !global arrays
   allocate(t_val(gsize))
else              !locate arrays
   allocate(t_val(dsize(i_part)))
endif
 . . .
 !concatonate local array to global array on partition 1
call concat_array(dsize,t_val)
```

`subroutine concat_array`(*dsize*, ***array, garray***)		
Arguments		Description
dsize(:)	⇒	(integer) number of values for each partition
array(:)	⇒	(integer, pointer) input (local array). Also output(global) if garray not present
garray(:)	⇒	(integer, pointer, optional) output array
array(:)	⇒	(real, pointer) input (local array). Also output(global) if garray not present
garray(:)	⇒	(real, pointer, optional) output array
array(:)	⇒	(double precision, pointer) input (local array). Also output(global) if garray not present
garray(:)	⇒	(double precision, pointer, optional) output array

`subroutine concat_array`(*nval, dsize*, ***array, garray***)		
Arguments		Description
nval	⇒	(integer) first index size (same on each partition)
dsize(:)	⇒	(integer) number of values for each partition
array(:,:)	⇒	(integer, pointer) input (local array). Also output(global) if garray not present
garray(:,:)	⇒	(integer, pointer, optional) output array
array(:,:)	⇒	(real, pointer) input (local array). Also output(global) if garray not present
garray(:,:)	⇒	(real, pointer, optional) output array
array(:,:)	⇒	(double precision, pointer) input (local array). Also output(global) if garray not present
garray(:,:)	⇒	(double precision, pointer, optional) output array

Table 18.34: Subroutine concat_array(): to take local array from each partition and concatenate into global array on partition=1.

18.4 User Programmable Routines

The set of user programmable routines (UPR):

```
subroutine upr_properties(...)
subroutine upr_generic(...)
subroutine upr_init(...)
subroutine upr_bnd_cond(...)
subroutine upr_sources(...)
```

all have a predefined interface (argument list) which will be detailed below. The calling sequence for these routines was previously illustrated in Section (18.2). Generally, each routine is called for each phase (species) and, in turn, for for each changeable variable (property, field etc.) . In order to restrict the user supplied routine to apply to specific phases/materials (species), conditional blocks (if/case-blocks) are normally used to surround 'variable-changing' code.

18.4.1 User properties routine

`subroutine upr_properties(`*pro_name*, *mat*, *iph*, *isp*, *icel1*, *icel2*, *jbnd1*, *jbnd2*, *jiu1*, *jiu2*, *jil1*, *jil2*, *fi_c*, *fi_b*, *fi_ui*, *fi_li*)

Arguments		Description
pro_name	⇒	(char), property name (see Table (18.36))
mat	⇒	(integer), material id
iph	⇒	(integer), phase id
isp	⇒	(integer), species id
icel1	⇒	(integer), starting cell index
icel2	⇒	(integer), ending cell index
jbn1	⇒	(integer), starting boundary face index
jbn2	⇒	(integer), ending boundary face index
jiu1	⇒	(integer), starting upper interface index
jiu2	⇒	(integer), ending upper interface index
jil1	⇒	(integer), starting lower interface index
jil2	⇒	(integer), ending lower interface index
fi_c(:)	⇒	(real, optional) scalar variable cell values
fi_b(:)	⇒	(real, optional) scalar boundary values
fi_ui(:)	⇒	(real, optional) scalar upper interface values
fi_li(:)	⇒	(real, optional) scalar lower interface values

Table 18.35: Subroutine upr_properties(): to modify properties.

The list of changeable properties (*pro_name*) are shown below.

The routine `upr_properties` is called for each material, phase and species. The availability of a given property can be tested for using the Fortran 95/2003 `present(...)` command. E.g.

	Cell	Boundary	Interface	Solid	Species	Phase
density	x	x	x	x	x	x
conductivity	x	x		x	x	x
specific_heat	x	x		x	x	x
gas_constant	x				x	x
lam_viscosity	x				x	x
lam_mass_diffusion	x	x			x	

Table 18.36: Changeable properties and types available.

if `prop_name` was "conductivity", then `present(fi_b)` would return false. Property values are typically modified by looping over the extents passed down, e.g. in Fortran 95/2003:

```
do ic = icel1, icel2
  fi_c(ic) = <some value>
end do
```

As this routine modifies properties of pure substances (which in general depend on the temperature and pressure) the user should use the `get_field()` routine to acquire the temperature and/or pressure field. If the argument `isp` is non-zero then the property for that species can be changed, otherwise only the phase property can be modified. See Section (18.7.1 for example).

18.4.2 User initialisation routine

`subroutine upr_init(`*var_name*, *idt*, *icel1*, *icel2*)		
Arguments		Description
var_name	⇒	(char), name of field that can be modified
idt	⇒	(integer), domain type id
icel1	⇒	(integer), starting cell id of material domain (for idt/var_name)
icel2	⇒	(integer), ending cell id of material domain (for idt/var_name)

Table 18.37: Subroutine upr_init(): to modify initial field.

Here, *var_name* can be one of:

```
velocity
pressure
temperature
turb_energy
dissipation
phase_vol_frac
spec_mass_frac
mass_frac_ps
```

For fluid phase equation *idt* is equal to the negative phase index, and for single-phase fluid, mixture of fluid phases or solid domain it coincides with the corresponding domain indices. In case of

species mass fraction equation idt is equal to species indices.
Typically, the user would specify the name of a phase or species object (e.g. steel) and the name of a variable (e.g. temperature) for which the corresponding cell values can be modified. The object id for this object name would then be retrieved via `get_id()`. A conditional block could then be written to test for the 'object id' and 'variable name' match. Inside this block a call to `get_field()` provides the current cell values via pointer arrays. These values can be now modified by looping over material domain cells according to the user specific criteria (see below for example, Section (18.7.2)).

18.4.3 User boundary conditions routine

`subroutine upr_bnd_cond`(*var_name*, *idt*, *ir*, *jbnd1*, *jbnd2*)

Arguments		Description
var_name	⇒	(char), name of field that can be modified
idt	⇒	(integer), domain type id
ir	⇒	(integer), boundary region id
icel1	⇒	(integer), starting boundary id for ir
icel2	⇒	(integer), ending boundary id for ir

Table 18.38: Subroutine upr_bnd_cond(): to modify boundary conditions.

This routine is called for each boundary region and each phase (species) present in the corresponding fluid/solid material domains (after the default initialisation is done by the solver). See Section (18.7.3 for example). The order in which the variables (*var_name*) are passed to this routine is the following: *temperature*, *heat_flux*, *velocity*, *pressure*, *turb_energy*, *dissipation*, *phase_vol_frac*, *spec_mass_frac* and *mass_frac_ps*. In order to modify specific variables, a conditional block block (e.g. case structure) should be set up. The domain type index *idt* is associated with the memory address of *var_name* field. For a fluid phase equation it is equal to the negative phase index, and for the single-phase fluid, mixture of fluid phases or solid domain it coincides with the corresponding domain indices. In case of species mass fraction equation *idt* is equal to species indices. In addition, the user needs to check that the current domain type index *idt* and boundary region *ir* is relevant. Note that a boundary condition type and corresponding options associated with the specified region should not be modified i.e. user accessible variables *l_reg_cond* and *l_reg_opts* available via *get_reg()* should not be modified. Also there is no need to select and modify region variables with the symmetry or outlet boundary condition type. The user specified static pressure must be relative to the reference pressure while the total (stagnation) pressure is always absolute.

Important note: In case of pressure, total pressure and mass flow boundary types the user has to supply turbulence intensity and length scale boundary fields instead of actual turbulence energy and dissipation fields, respectively. They are temporarily stored in the same arrays used for the turbulence energy and dissipation.

subroutine upr_sources(*mat*, *ieq*, *idt*, *iph*, *isp*, *icel1*, *icel2*, *vol_ph*, *ap_fi*, *src_fi*)

Arguments		Description
mat	⇒	(char), index of (fluid/solid) material domain
ieq	⇒	(integer), equation index
idt	⇒	(integer), domain type index
iph	⇒	(integer), fluid/solid phase index
isp	⇒	(integer), fluid species index
icel1	⇒	(integer), starting boundary id for ir
icel2	⇒	(integer), ending boundary id for ir
vol_ph	⇒	(real), cell volume occupied by phase
ap_fi	⇒	(real,output), central coefficient of the discretised equation
src_fi	⇒	(real,output), source term of the discretised equation

Table 18.39: Subroutine upr_sources(): to add source terms to equations.

18.4.4 User sources routine

This routine is called every outer iteration (in the context of SIMPLE algorithm) for each phase (species) transport equation which is solved for. Here *idt* is the domain type index associated with the memory address of equation field. For fluid phase equation it is equal to the negative phase index. For single-phase fluid, mixture of fluid phases or solid domain it coincides with the corresponding domain indices. In case of species mass fraction equation *idt* is the *idt* of the parent phase.

This routine is used to add source terms either explicitly by adding to `src_fi` or implicitly by adding to `ap` (the central coefficient). For a user selected phase or species object, a conditional test is required to match *isp* or *iph* with the corresponding user object id (via `get_id(...)`).
Note it is not possible to modify the source term for the selected phase and for more than one equation at the same time. In case of a solid material the phase name should be the same as material domain name.
Depending on the nature of the additional source, the user might need to declare a pointer array for the variable whose source need to be modified and also additional pointer arrays if other variables are required (for example the density is often required, or if the source of turbulence energy is modified the dissipation rate may be needed). Then all required fields (cell values) can be obtained by using the `get_field()` access routine.
In general, additional sources can be expressed, linearly, by:

```
src = src_1 -src_2*fi
```

where `src_1` is the phi(fi) independent term, and `src_2*fi` is phi-dependent part. In order to improve stability, `src_2` can be added to the source (explicit) or to `ap` (implicit) depending on to the sign:

```
src_fi(1,ic)=src_fi(1,ic)-min(0.,src_2)*fi(ic)
ap_fi(ic)=ap_fi(ic)+max(0.,src_2)
```

For the phi-independent part (`src_1`), if it is negative (always), it can be added to `ap` by artificially introducing a dependence on phi by scaling it with `1./fi_old(ic)` from the previous iteration.

```
src_fi(1,ic)=src_fi(1,ic)+max(0.,src_1)
ap_fi(ic)=ap_fi(ic)-min(0.,src_1)/fi(ic)
```

See Section (18.7.4) for example.
In case of momentum equations all velocity components have the same central coefficient ap: ap_u()=ap_v()=ap_w(). Therefore, the user should try to redefine negative source term -src_2 to be equal for all velocity components. If this is not possible, the negative source term(s) are added to src_fi().
Note: The source terms used above (src, src_1, src_2) must have the units (physical dimensions) equal to the units of mass flow rate [kg/sec] times the units of variable phi. For example, the units will be:

```
mass, mass/volume fraction source   => [kg/sec]*[1]        =[kg/sec]
momentum (velocity) source          => [kg/sec]*[m/sec]    =[N]
total enthalpy/energy source        => [kg/sec]*[J/kg]     =[W]
turbulent kinetic energy source     => [kg/sec]*[(m/sec)^2]=[W]
turbulent energy dissipation source => [kg/sec]*[m^2/sec^3]=[W/s]
```

As the cell volume occupied by phase vol_ph() is available the user will have to evaluate first the source terms per unit volume (or per unit mass times density) and multiply it by phase volume vol_ph(). In case of single phase fluid or multi-phase mixture model the phase cell volume is the actual cell volume.

18.4.5 User generic routine

`subroutine upr_generic(`*id*, *icall_pos*`)`

Arguments		Description
id	⇒	(integer), global id
icall_pos	⇒	(integer), calling position

Table 18.40: Subroutine upr_generic(): to modify variables in arbitrary fashion.

This routine is used to modify variables in a general way. It is called at various positions during the simulation (see Table (18.41)). See Section (18.7.5 for example).

18.5 Writing and Compiling UPR; Dynamic Shared Objects

Currently, UPRs must be written in Fortran 95/2003. The subroutine names and argument list must adhere to those listed in Section (18.4). In addition, the Fortran 95/2003 module “upr.mod” must

CALL ID	PROGRAM POSITION
icp_beg_run	start of simulation, after initialisation
icp_beg_time	beginning of each time step
icp_end_iter	end of each iteration
icp_end_time	end of each time step
icp_end_run	end of simulation

Table 18.41: Program position identifiers.

be included to provide access to the user accessible routines/variables (UARs/UAVs). Compilation of UPRs can be done by running the "umake" utility, e.g.

```
umake upr_bound.f90
```

which would create a shared object ("libupr.so"). Multiple source files can be specified on the command line. Options to umake include:

`-f`	name of compiler program (default: ifort)
`-compver`	compiler version
`-32`	run in 32-bit mode
`-64`	run in 64-bit mode
`-o`	output file name (default: libupr.so)

Current compiler support is Intel & gfortran (GNU). Compiler/linker flags can be set via the `FFLAGS` and `LDFLAGS` environment variables.

18.6 UPR Check/Report Messages

Error messages generated during the compilation of a UPR typically arise from a missing `use upr` statement or some inconsistency in the way a routine (UAR) is called.
Any error messages generated during the running of an UPR are usually quite self-explanatory. Typically error/warning messages may arise from trying to retrieve (or set) non-available fields/-properties for a given material/phase etc.
NOTE: Certain arrays that are directly accessible should only be read. For example, the boundary conditions array can be retrieved (via a Fortran 95/2003 pointer) and modified (illegally), causing unexpected results. Currently, there are no means of enforcing read-only arrays when read via a pointer.

18.7 Examples

18.7.1 Example of the user properties routine

```
!==========================================================================
!Description       : User programming routine to specify thermo-physical
!                    properties of individual phase or species.
```

```
!Filename          : upr_properties.f90
!===============================================================================

!----------------------------------------------------------------------------
subroutine upr_properties(pro_name,mat,iph,isp,icel1,icel2,jbnd1,jbnd2 &
                         ,jiu1,jiu2,jil1,jil2,fi_c,fi_b,fi_ui,fi_li)
!This routine is called for a phase iph (species isp) present in fluid/solid
!----------------------------------------------------------------------------
!Modules used (imported type definitions, parameters, scalars and arrays):
use upr
implicit none
character(len=*),intent(in):: pro_name           !name of a property field
integer(iwp),intent(in)    :: mat               &!index of material domain
                             ,iph               &!fluid/solid phase index
                             ,isp               &!fluid species index
                             ,icel1,icel2       &!start & end cell indices
                             ,jbnd1,jbnd2       &!start & end bnd face indices
                             ,jiu1,jiu2         &!start & end high interfaces
                             ,jil1,jil2          !start & end low interfaces
real(wph),intent(inout)    :: fi_c(icel1:icel2) &!property's cell values
                             ,fi_b(jbnd1:jbnd2) &!boundary values
                             ,fi_ui(jiu1:jiu2)  &!values at upper interfaces
                             ,fi_li(jil1:jil2)   !values at low interfaces
optional                   :: fi_b,fi_ui,fi_li

!Local Variables
real(wph),pointer          :: t(:)   => null()  &!cell temperature
                             ,tb(:)  => null()  &!bnd temperature
                             ,ti(:)  => null()  &!uper interface temperature
                             ,tli(:) => null()   !low interface temperature

real(wph),pointer          :: p(:)   => null()  &!cell pressure
                             ,pb(:)  => null()  &!bnd pressure
                             ,pi(:)  => null()  &!upper interface pressure
                             ,pli(:) => null()   !low interface pressure

real(wp),pointer           :: p_ref(:)=>null()  &!mat domain reference press
                             ,rph_val(:,:)=>null()!reference phase propty vals

integer(iwp),pointer       :: n_li(:) =>null()  &!number of low interfaces
                             ,ia_li(:)=>null()  &!start address for low intf
                             ,l_li(:) =>null()   !list of low interfaces
real(wph)                  :: gc                 !gas constant

character(len=len_var_name):: usr_obj_name   !user selected name for either
                                             !phase or species object

integer(iwp)               :: iget          &!flag to get phase/species index
                             ,iobj          &!phase or species (object) index
                             ,ipro          &!index of property
                             ,usr_obj_id     !index of phase or species object
                                             !corresponding to upr_obj_name
integer(iwp)               :: ide           &!domain type index for energy
                             ,idp           &!domain type index for pressure
                             ,ieq           &!equation index
                             ,ic            &!cell index
                             ,jb            &!bnd face index
                             ,jl,ji          !material interface indices
character(len=len_var_name):: var_name       !variable field name

  usr_obj_name=' '

  !Comment line below to diasble this routine
  usr_obj_name='air'
  if (usr_obj_name == ' ') return

  !Find if the phase or species object is passed to
  if (isp > 0) then
    iobj=isp
    iget=iget_specs
```

```
else
  iobj=iph
  iget=iget_phase
end if
!Get associated object id with this object ("usr_obj_name")
call get_id(usr_obj_name,usr_obj_id,iget)

!Only perform block below if have right object id
if (iobj == usr_obj_id) then

  !Get property index
  !call get_id(pro_name,ipro,iget_ph_pro)

  !Determine property to be calculated for
  select case (pro_name)
    case ('density')
      !upr -> start
      !Calculate density for ideal non-isotropic gas flow
      !Pressure field is required
      call get_id('mass_pressure',ieq,iget_eq) !Get index of pressure eq
      idp=eq_idt(ieq,iph,iget_phase)           !Get pressure domain index
      !Get pressure field (NOTE: returned are RELATIVE pressure values)
      var_name='pressure'
      call get_field(idp,scal_field,var_name &
                    ,fi_c=p,fi_b=pb,fi_ui=pi,fi_li=pli)
      !Get reference pressure
      call get_mat('r_pressure',p_ref)

      !Get phase molecular weight and calculate gas constant
      call get_property('r_phase_values',rph_val)
      gc=8315.0_wp/rph_val(imolw,iobj)

      !If temperature field is required
      call get_id('energy',ieq,iget_eq) !Get index of energy equation
      ide=eq_idt(ieq,iph,iget_phase)    !Get energy (temp) domain type
      !Get temperature field
      var_name='temperature'
      call get_field(ide,scal_field,var_name &
                    ,fi_c=t,fi_b=tb,fi_ui=ti,fi_li=tli)

      !Define property values as a function of temperature, pressure or both
      !temperature and pressure : fi_pro=f(T,p_absolute)
      do ic=icel1,icel2
        fi_c(ic)=(p_ref(mat)+p(ic))/(gc*t(ic))
      end do

      !Boundary values
      if (present(fi_b)) then
        do jb=jbnd1,jbnd2
          fi_b(jb)=(p_ref(mat)+pb(jb))/(gc*tb(jb))
        end do
      end if

      !Upper interface values
      if (present(fi_ui)) then
        do ji=jiu1,jiu2
          fi_ui(ji)=(p_ref(mat)+pi(ji))/(gc*ti(ji))
        end do
      end if

      if (present(fi_li)) then
        !To define low interface values we need a number of low interfaces
        !n_li(mat), initial address ia_li(mat) & the list of low interfaces
        !l_li()
        call get_mat('n_low_interface',n_li)
        call get_mat('isa_low_interface',ia_li)
        call get_mat('l_low_interface',l_li)
        do jl=1,n_li(mat)
          ji=l_li(ia_li(mat)+jl)
          fi_li(ji)=(p_ref(mat)+pli(ji))/(gc*tli(ji))
          end do
          nullify (n_li,ia_li,l_li)
        end if
        nullify (p,pb,pi,pli)
        nullify (p_ref,rph_val)
        nullify (t,tb,ti,tli)
        !
      case ('lam_viscosity')
        !Boundary and interface values are not modifiable.
        !If temperature field is required
        call get_id('energy',ieq,iget_eq) !Get index of energy equation
        ide=eq_idt(ieq,iph,iget_phase)    !Get energy (temp) domain type
        !Get temperature field
        var_name='temperature'
        call get_field(ide,scal_field,var_name &
                      ,fi_c=t,fi_b=tb,fi_ui=ti,fi_li=tli)

        !Define property values as a function of temperature, pressure or both
        !temperature and pressure : fi_pro=f(T,p_absolute)
        do ic=icel1,icel2
          fi_c(ic)=1.716e-5_wp*(t(ic)/273.15)**1.5*(273.15+111.)/(t(ic)+111.)
        end do
        nullify (t,tb,ti,tli)
        !
      case ('conductivity')
      case ('specific_heat')
      case ('gas_constant')
      case ('lam_mass_diffusion')
    end select
    !
    !upr -> end
    !
  end if
  !
  !Next 'usr_obj_name' and select case(pro_name): copy & edit the above code
  !
end subroutine upr_properties
```

The above example is generalised to handle any multi-species/phase (and multi-domain for that matter) case. E.g. for the case where there are 2 phases & 4 species (air+methane, water+methane) this case will only set properties for the "air"-species component. To set properties for more than one component/phase repeat code block and change `usr_obj_name` accordingly. The properties set (using the `case` construct) are density and laminar viscosity. Within the "density"case block the temperature & pressure fields are retrieved from the solver, along with the molecular weight. Prior to calling `get_field`, it is first necessary to determine the domain type id (idt). This is done via the 2 calls:

```
call get_id('mass_pressure',ieq,iget_eq) !Get equation index (ieq) from equation name
                                         !and object type (iget_ieq)
idp=eq_idt(ieq,iph,iget_phase)           !Get domain type id from ieq and phase id (iph)
                                         !and object type (iget_phase)
```

The object type passed to `get_id` is the transport equation identifier `iget_eq` (see Table (18.23). An alternative way of determining `ieq` via `get_id`, is to simply look up the identifier as listed in Table (18.5) (`global data identifiers for transport equations`). The call to `eq_idt` returns the domain type id (domain or phase) from the equation identifier `iget_eq`, phase id `iph` and the object type `iget_phase`. The object type (argument 3) sets the context for the object id (argument 2). If the domain type id doesn't exist `eq_idt` returns zero, e.g. for a 2-domain (2-material) fluid-solid case,

```
idt=eq_idt(ifmom,2,iget_mat)              !Get domain type id from ieq and phase id (iph)
```

would return zero as momentum equation is not solved on material 2 (solid). The `var_name` passed to `get_field` can be any one of those listed in Table (18.26). There are two ways of calling `get_field` (see Table (18.27)):

```
get_field(idt,iget,var_name,fi)
get_field(idt,rank,var_name,fi_c,fi_b,fi_ui,fi_li,fi_c_o,fi_c_oo,fi_f,fi_n,indx)
```

The first method is for retrieving a single field of specified type (`iget`), e.g. cell or boundary etc. (see Table (18.27) for valid codes). The second calling method allows the user to optionally pull back many field types in one go. If the variable (`var_name`) is a scalar, then `rank` should be rank-1 array (use parameter scal_field). If variable is a vector, then `rank` should be rank-2 array (use parameter vect_field). As all the field types are optional, and some can be missing, it is recommended to used named arguments, e.g.

```
call get_field(idt,vect_field,'velocity',fi_c=v,fi_c_o=vold)
```

would return velocity & velocity-old (from previous time step) at cell centres into user defined pointers `v` & `vold` respectively. The preceding example pulls back pressure values for cells (p), boundaries (pb), upper (pi) & lower (pli) material interfaces. This is followed by calls to get the reference pressure and molecular weight. The call to:

```
get_property('rphase_values',rph_val)
```

returns the property array in `rph_val` which can be indexed using the property id's as listed in Table (18.5). Once the temperature is found, the density at the cells are modified thus:

```
do ic=icell,icel2
  fi_c(ic)=(p_ref(mat)+p(ic))/(gc*t(ic))
end do
```

Where `icel1` & `icel2`, which are passed down into this routine, represent the starting and ending cell index for this material domain (`mat`). The following 3 if-blocks (for this density case-block) modify the boundary and material interface values in a similar way. The boundary and interface arrays are optional passed down depending on the variable. Property boundary values, only density, conductivity, specific heat capacity & laminar mass diffusion (multi-species) can be changed. For material interface values, only the density can be modified.

18.7.2 Example of the user initialisation routine

```
!=============================================================================
!Description       : User programming routine to specify initial cell values
!                    for VECTIS 4 solution.
!==============================================================================

!---------------------------------------------------------------------------
subroutine upr_init(var_name,idt,icel1,icel2)
!This routine is called for each phase (species) present in fluid/solid
!material domains (after the default initialisation is done by the solver).
!---------------------------------------------------------------------------
!Modules used (imported type definitions, parameters, scalars and arrays):
use upr
implicit none
integer(iwp),intent(in)    :: idt          &!domain type index
                             ,icel1        &!material domain start cell index
                             ,icel2         !material domain end cell index
character(len=*),intent(in):: var_name      !name of a variable field

!Local Variables
real(wph),pointer :: fi(:)=>null()          !pointer array for a scalar field
real(wph),pointer :: fi_vec(:,:)=>null()    !pointer array for a vector field

character(len=len_var_name):: usr_obj_name &!user selected name for either
                                            !phase or species object
                             ,usr_var_name  !user selected variable name
integer(iwp)               :: iget         &!flag to get phase/species/PS index
                             ,usr_obj_id   &!index of phase, species or PS object
                                            !corresponding to upr_obj_name
                             ,ic            !cell index
real(wph),pointer :: xcell(:,:)=>null()     !cell centre coordinates

  usr_obj_name='aluminium'
  usr_var_name='temperature '

  if (usr_obj_name == ' ' .or. usr_var_name == ' ') return
  !
  iget=iget_phase
  if (var_name == 'spec_mass_frac') iget=iget_specs
  if (var_name == 'mass_frac_ps') iget=iget_ps
  call get_id(usr_obj_name,usr_obj_id,iget)

  if (abs(idt) == usr_obj_id .and. var_name == usr_var_name) then
    if (var_name == 'velocity') then
      call get_field(idt,iget_cell,var_name,fi_vec)
      !upr -> start
      do ic=icel1,icel2
      end do
      !upr -> end
      fi_vec => null()
    else
      call get_field(idt,iget_cell,var_name,fi)
      !upr -> start
      call get_grid_geom('xyz_cell_c',xcell)
      do ic=icel1,icel2
        fi(ic)=xcell(1,ic)*xcell(2,ic)*xcell(3,ic)
      end do
      !upr -> end
      fi => null()
      xcell => null()
    endif
  end if
  !
  !Next 'usr_obj_name' and 'usr_var_name': copy & edit the above code
  !
end subroutine upr_init
```

18.7.3 Example of the user boundary conditions routine

```
!==============================================================================
!Description       : User programming routine to specify boundary values
!                    (profiles) for VECTIS 4 solution  variables.
!==============================================================================

!----------------------------------------------------------------------------
subroutine upr_bnd_cond(var_name,idt,ir,jbnd1,jbnd2)
!This routine is called for each boundary region and each phase (species)
!present in the corresponding fluid/solid material domains
!(after the default initialisation is done by the solver).
!----------------------------------------------------------------------------

!Modules used (imported type definitions, parameters, scalars and arrays):
use upr
implicit none
integer(iwp),intent(in)     :: idt          &!domain type index
                             ,ir            &!boundary region index
                             ,jbnd1         &!starting bnd region face
                             ,jbnd2          !ending bnd region face
character(len=*),intent(in):: var_name       !name of a variable field

!Local Variables
real(wph),pointer    :: fib(:)=>null()      !pointer array for a scalar field
real(wph),pointer    :: fib_vec(:,:)=>null()!pointer array for a vector field
integer(iwp),pointer :: l_bc(:,:)=>null()   !pointer array for bnd conditions

character(len=len_var_name):: usr_obj_name &!user selected name for either
                                             !phase or species object
                             ,usr_reg_name  !user selected variable name
integer(iwp)                :: iget         &!flag to get phase/species index
                             ,usr_obj_id    &!index of phase or species object
                                             !corresponding to upr_obj_name
                             ,usr_reg_id    &!index of user selected reg
                             ,jb            &!boundary face index
                             ,ic             !cell index
real(wph),pointer :: xbnd(:,:)=>null()       !bnd face centre coordinates

!upr => Diamond case
real(wph),pointer :: ub(:,:)   =>null()     &!bnd face velocity
                    ,teb(:)    =>null()      !bnd turbulent kinetic energy
real(wp),pointer  :: reg_bval(:,:)=>null()   !region-wise var bnd values
real(wp)                    :: zinl(0:50)   &!coordinate data for the profile
                             ,uz  (0:50)    &!velocity profile
                             ,tez (0:50)    &!turbulent energy profile
                             ,u2z,v2z       &!normal turbulent stresses
                             ,uvz           &!shear uv stress
                             ,zfac          &!interpolation factor
                             ,uzk           &!inerpolated bnd velocity
                             ,tezk           !interpolated turb energy
integer(iwp)                :: k            &!loop index
                             ,idm           &!domain type index for momentum
                             ,ndata         &!number of data
                             ,ifun           !file unit for reading bnd cond


  !Do nothing for a symmetry or outlet boundary type
  call get_reg('l_reg_cond',l_bc) !get a list of bnd conditions
  if (l_bc(ibct,ir) == ibsym .or. l_bc(ibct,ir) == ibout) return

  !Set object name & boundary interested in
  usr_obj_name='air'
  usr_reg_name='Inlet'

  if (usr_obj_name == ' ' .or. usr_reg_name == ' ') return
  !
  !Get a phase or species index usr_obj_id for the selected usr_obj_name
  iget=iget_phase
```

```
if (var_name == 'spec_mass_frac') iget=iget_specs
if (var_name == 'mass_frac_ps') iget=iget_ps
call get_id(usr_obj_name,usr_obj_id,iget)

!Get a region index usr_reg_id for the selected region name usr_reg_name
call get_id(usr_reg_name,usr_reg_id,iget_breg)

if (abs(idt) == usr_obj_id .and. ir == usr_reg_id) then
  !
  select case (var_name)

    case ('temperature')
      call get_field(idt,iget_bnd,var_name,fib)
      !upr -> start
      call get_grid_geom('xyz_bndf_c',xbnd)
      do jb=jbnd1,jbnd2
        !fib(jb)=xbnd(1,jb)*xbnd(2,jb)*xbnd(3,jb)
      end do
      xbnd => null()
      !upr -> end
      fib => null()

    case ('heat_flux')
      call get_field(idt,iget_bnd,var_name,fib)
      !upr -> start
      do jb=jbnd1,jbnd2
      end do
      !upr -> end
      fib => null()

    case ('velocity')
      call get_field(idt,iget_bnd,var_name,fib_vec)
      !upr -> start
      do jb=jbnd1,jbnd2
      end do
      !upr -> end
      fib_vec => null()

    case ('pressure')
      !The user specified static pressure must be relative to the reference
      !pressure ref_press while the total (stagnation) pressure is always
      !absolute.
      call get_field(idt,iget_bnd,var_name,fib)
      !upr -> start
      do jb=jbnd1,jbnd2
      end do
      !upr -> end
      fib => null()


    case ('turb_energy')
      !In case of pressure, total and mass flow boundary types the user
      !has to supply turbulence intensity instead of actual turbulence
      !energy. The turbulence intensity is also saved in the array fib.

      !Initialise velocity and turbulence at inlet from tabulated
      !experimental data for the boundary layer profile (k, z associated
      !with the z-axis == xbnd(3,:))
      !Get access to fields
      call get_field(idt,iget_bnd,'turb_energy',teb) !turbulent energy
      !Get equation index
      idm=eq_idt(ifmom,usr_obj_id,iget_phase)
      call get_field(idm,iget_bnd,'velocity',ub)     !velocity
      !Get bnd face centre coordinates
      call get_grid_geom('xyz_bndf_c',xbnd)
      !Get region-wise uniform bnd values
      call get_reg('phase_value',reg_bval)

      zinl(:)=zero
      uz(:)=zero
    tez(:)=zero
    ifun=0
    call get_funit(ifun)
    call open_file (ifu=ifun,fln='wing.inl' &    !module file_utilities
        ,st='old',acc='sequential',fm='formatted')
    read(ifun,*)
    read(ifun,*) ndata
    do k=1,ndata
      read(ifun,*) zinl(k),uz(k),u2z,v2z,uvz
      tez(k)=0.5*(u2z+v2z+2.*v2z-u2z)*reg_bval(1,ir)**2
    end do

    do jb=jbnd1,jbnd2
      k=ndata
      if (xbnd(3,jb) >=zinl(k)) then
        ub(1:3,jb)=uz(k)*reg_bval(1:3,ir)
        teb(jb)=tez(k)
      else
        do k=1,ndata
          if (xbnd(3,jb) < zinl(k) .and. xbnd(3,jb) >= zinl(k-1)) then
            zfac=(xbnd(3,jb)-zinl(k-1))/(zinl(k)-zinl(k-1))
            uzk=uz(k-1)*(1.-zfac)+uz(k)*zfac
            tezk=tez(k-1)*(1.-zfac)+tez(k)*zfac
            ub(1:3,jb)=uzk*reg_bval(1:3,ir)
            teb(jb)=tezk
            exit
          end if
        end do
      end if
      !edb(jb)=cmu(mat)*rproph(idens,ir)*ttur(idt)%teb(jb)**2 &
      !         /(rproph(ivis,ir)*rbc(ibturls,ir))
    end do
    close(ifun)
    nullify (ub,teb,xbnd,reg_bval)

  case ('dissipation')
    !In case of pressure, total and mass flow boundary types the user
    !has to supply turbulence length scale instead of actual dissipation.
    !The dissipation is also saved in the array fib.
    call get_field(idt,iget_bnd,var_name,fib)
    !upr -> start
    do jb=jbnd1,jbnd2
    end do
    !upr -> end
    fib => null()

  case ('phase_vol_frac')
    call get_field(idt,iget_bnd,var_name,fib)
    !upr -> start
    do jb=jbnd1,jbnd2
    end do
    !upr -> end
    fib => null()

  case ('spec_mass_frac')
    call get_field(idt,iget_bnd,var_name,fib)
    !upr -> start
    do jb=jbnd1,jbnd2
    end do
    !upr -> end
    fib => null()
  case ('mass_frac_ps')
    call get_field(idt,iget_bnd,var_name,fib)
    !upr -> start
    do jb=jbnd1,jbnd2
    end do
    !upr -> end
    fib => null()
  !
end select
end if
!
!Next 'usr_obj_name' and 'usr_reg_name': copy & edit the above code
!
end subroutine upr_bnd_cond
```

18.7.4 Example of the user sources routine

```
!==============================================================================
!Description       : User programming routine to specify additional sources for
!                    the transport equations solved by Vectis 4.
!==============================================================================

!---------------------------------------------------------------------------
subroutine upr_sources(mat,ieq,idt,iph,isp,icel1,icel2,vol_ph,ap_fi,src_fi)
!
!This routine is called every outer iteration (in the context of SIMPLE
!algorithm) for each phase (species) transport equation which is solved for.
!---------------------------------------------------------------------------

!Modules used (imported type definitions, parameters, scalars and arrays):
use upr
implicit none
integer(iwp),intent(in) :: mat                  &!index of material domain
                          ,ieq                  &!equation index
                          ,idt                  &!domain type index
                          ,iph                  &!fluid/solid phase index
                          ,isp                  &!fluid species index
                          ,icel1,icel2           !start & end cell indices
real(wph),intent(in)    :: vol_ph(icel1:icel2)   !cell volume*volume_fraction
real(wph),intent(inout) :: ap_fi(icel1:icel2)   &!equation central coeff.
                          ,src_fi(1:3,icel1:icel2)!equation source term(s)

!Local Variables
!Scalar or velocity field corresponding to the equation index ieq
real(wph),pointer       :: fi(:)   => null()     !cell values of scalar
real(wph),pointer       :: u(:,:)  => null()     !cell values of velocity

!Declare other pointers as required
real(wph),pointer       :: den(:)  => null()     !density field


character(len=         &
           len_var_name):: usr_obj_name         &!user selected name for
                                                 !either phase or species
                          ,eq_name              &!name of transport equation
                          ,var_name              !name of variable field

integer(iwp)            :: iobj                 &!phase/species object indx
                          ,usr_obj_id            !index of phase/species object
                                                 !corresponding to usr_obj_name
integer(iwp)            :: ic                    !cell index
real(wph)               :: src                  &!calculated scalar src term
                          ,src_vel(1:3)          !calculated momentum src term
real(wph)               :: fi_old
real(wph)               :: tol
real(wph),pointer       :: xc(:,:)  => null()    !cell centres

  !upr -> start
  !Select a phase (species) name
  usr_obj_name=' '
  !upr -> end

  !usr_obj_name='air'
  if (usr_obj_name == ' ') return

  !Find phase (species) index from the selected  'usr_obj_name'
```

```
    if (isp > 0) then
      iobj=isp
      call get_id(usr_obj_name,usr_obj_id,iget_specs)
    else
      iobj=iph
      call get_id(usr_obj_name,usr_obj_id,iget_phase)
    end if

    if (iobj == usr_obj_id) then
      !Default values for sources
      src_vel(:)=0.; src=0.
      !Get equation name
      call get_name(ieq,eq_name,iget_eq)

      select case (eq_name)
        !
        case ('momentum')
          !upr -> start
          !Get velocity field if required
          var_name='velocity'
          call get_field(idt,iget_cell,var_name,u)
          !Define the source term vector src_vel(1:3) per unit volume
          !and if appropriate a negative source term 'src' per unit volume
          !equal for all velocity components (u,v,w): src=src_u=src_v=src_w
          do ic=icel1,icel2
            !src=
            !src_vel(1)=
            !src_vel(2)=
            !src_vel(3)=
            src_fi(1:3,ic)=src_fi(1:3,ic)+src_vel(1:3)*vol_ph(ic)
            ap_fi(ic)=ap_fi(ic)+src*vol_ph(ic)
          end do
          nullify (u)
          !upr -> end

        case ('mass_pressure')
          !
        case ('energy')
          !upr -> start
          !Get temperature field if required
          var_name='temperature'
          call get_field(idt,iget_cell,var_name,fi)
          call get_grid_geom('xyz_cell_c',xc)
          do ic=icel1,icel2
            !Define the source term(s) 'src' per unit volume
            !src=
            tol=0.05
            if (xc(1,ic)<-2.+tol .and. xc(1,ic)>-2.-tol .and. &
                xc(2,ic)<3.5+tol .and. xc(2,ic)>3.5-tol) then
            else
              cycle
            end if
            src=-1.e6
            fi_old=fi(ic)+sign(small,fi(ic))
            src=src*vol_ph(ic)
            src_fi(1,ic)=src_fi(1,ic)+max(0.,src)
            ap_fi(ic)=ap_fi(ic)-min(0.,src)/(fi_old)
          end do
          nullify (fi)
          !upr -> end

      end select
      !
    end if
    !
    !Next 'usr_obj_name' and select case(eq_name): copy & edit the above code
    !
  end subroutine upr_sources
```

This routine sets sources for various equations. Equation name is derived for `ieq` via the call to `get_name`. The phi-dependent part of the source is added to the source or to `ap` according to the source sign.

18.7.5 Example of the user generic routine

This example is used to write a temperature line-out (at x=0.5) to a file at the end of each iteration. The `icp_beg_run` case block is used to determine the total size (over the parallel partitions) of the array to store the values. Then the global y-position array is determined for the first partition 1 (via the use of `concat_array`). The `icp_end_iter` proceeds similarly, storing the temperature on each partition and the concatenating onto partition 1. Partition 1 has the sole responsibility of writing the file (y .vs. t).

```
!---------------------------------------------------------------------------
subroutine upr_generic(id,icall_pos)
!This general, model independent routine is called from several positions in
!the solution cycle.
!---------------------------------------------------------------------------
use upr
implicit none
! Global (public) declarations (types, parameters, scalars, arrays)
integer(iwp),intent(in) :: id         &!global solution domain id (==1)
                          ,icall_pos   !calling position flag in the solution
                                       !cycle
integer(iwp),pointer                :: iscd(:)  &!integer pointers
                                      ,iecd(:)  !integer pointers

real(wp),pointer      :: xcc(:,:) &!cell centre coords
                      ,tfield(:)& !cell temperature
                      ,ufield(:,:)!cell velocity vector
integer(iwp) ::        ic&        !cell counter
                      ,idt&       !solution variable index
                      ,iup1=21&   !file unit
                      ,idom&      !domain counter,
                      ,n_dom&     !number of fluid/solid domains
                      ,i_part&    !this partitions number
                      ,n_part&    !number of paritions
                      ,n          !counter for compressed arrays
integer(iwp),allocatable,save :: dsize(:)        !number of stored values on each partition
integer(iwp),save :: gsize                       !total number of stored values
real(wph),allocatable,save    :: xval(:),tval(:) !compressed arrays for y-coordinate and temperature
character(len=80)             ::filename&        !output filename
                               ,proj_name&       !project name
                               ,proj_num         !project number

!write temperature profile at x=0.5
!For parallel cases the values are calculated on each partitions and then a
!concatonated list of the data from all partitions is formed on partition 1 and then
!only partition 1 writes the output. For efficient data exchange a compressed list of values
!is exchanged.

  select case (icall_pos)
    case (icp_beg_run,icp_end_iter)
      n_part=get_number('n_partitions')
      i_part=get_number('n_current_part')
      n_dom=get_number('n_domains')
      call get_domain('ise_cell', iscd,iecd)
      call get_grid_geom('xyz_cell_c',xcc)
  end select

  select case (icall_pos)
    case (icp_beg_run)
    !beginning of simulation
    !setup data storage and generate compressed data arrays

    !Find size of compressed data array
    n=0
    do idom=1,n_dom
      do ic=iscd(idom),iecd(idom)
        if ( abs(xcc(1,ic)-0.5E0) < 2.0E-3 ) n=n+1
      enddo
    enddo
    allocate(dsize(n_part))
    dsize=0
    dsize(i_part)=n !number per partition
    call global_sum(dsize,n_part)
    gsize=sum(dsize(:)) !total number

    !allocate memory for compressed arrays of of y-coordinate and temperature.
    ! On partition 1 globally sized arrays are created.
    if(i_part==1)then !global arrays
      allocate(tval(gsize))
      allocate(xval(gsize))
    else !locate arrays
      allocate(tval(dsize(i_part)))
      allocate(xval(dsize(i_part)))
    endif
    n=0
    !store y-coordinate values
    do idom=1,n_dom
      do ic=iscd(idom),iecd(idom)
        if ( abs(xcc(1,ic)-0.5E0) < 2.0E-3 ) then
          n=n+1
          xval(n)=xcc(2,ic)
        endif
      enddo
    enddo
    call concat_array(dsize,xval) !concatonate local arrays to global array on partition 1
    if(i_part>1) deallocate(xval) !free local arrays for y-coordinate

  case (icp_beg_time)
    !beginning of each time step
  case (icp_end_iter)
    !end of each iteration

    !store temperature value for each partition
    n=0
    do idom=1,n_dom
      idt=eq_idt(iene,idom,iget_dom)
      call get_field(idt,iget_cell,'temperature',tfield)
      do ic=iscd(idom),iecd(idom)
        if ( abs(xcc(1,ic)-0.5E0) < 2.0E-3 ) then
          n=n+1
          tval(n)=tfield(ic)
        endif
      enddo
    enddo
    call concat_array(dsize,tval) !concatonate local arrays to global array on partition 1

    !partition 1 writes data to file
    if(i_part==1)then
      call get_run_ctrl('project_name',proj_name)
      call get_run_ctrl('proj_run_number',proj_num)
      filename=proj_name(1:len_trim(proj_name))//'.UPR'//proj_num
      open (unit=iup1, file=filename,status='replace' &
                  ,access='sequential',form='formatted')
      do n=1,gsize
        write(iup1,'(2e18.8)') xval(n), tval(n)
      enddo
      close(iup1)
    endif
    case (icp_end_time)
      !end of timestep
    case (icp_end_run)
      !end of simulation
      !free memory
      deallocate(tval)
      deallocate(dsize)
      if(i_part==1) deallocate(xval)

  end select
  !
end subroutine upr_generic
```

19 TUTORIALS

These tutorials contain simple examples chosen to help demonstrate the typical usage of the software whilst attempting to introduce the user to some of the many numerous features available.

The latest version of the VECTIS-MAX manuals and tutorials can be found online on the Ricardo Software website: `http://www.software.ricardo.com/support/manuals/vectismax`

Check here for updates and also PDF and XHTML versions.

19.1 Basic Tutorial

Get the necessary files for the port flow tutorial

`http://www.software.ricardo.com/support/tutorials/vectismax/TutorialFiles Basic`

19.1.1 VECTIS Workflow

The VECTIS workflow consists of several stages illustrated in Figure 19.1

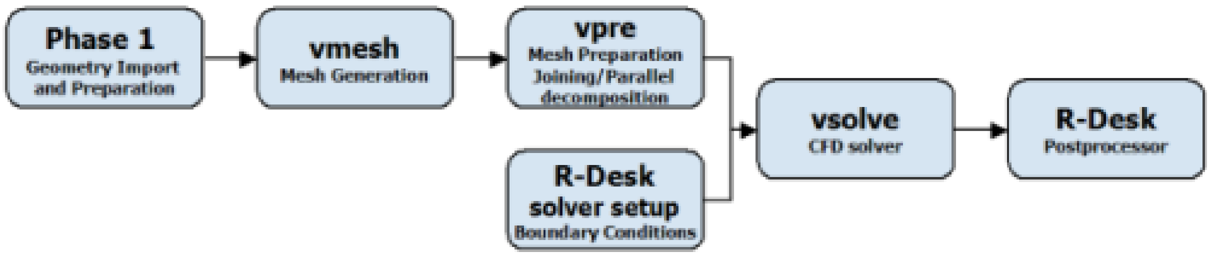


Figure 19.1: VECTIS workflow

- Geometry Import and Preparation (Phase 1)
- Mesh Generation (vmesh)

- Mesh Preparation (vpre)
- Solver Setup (R-Desk)
- CFD Solver (vsolve)
- Post-processing (R-Desk)

19.1.2 VECTIS Structure

Vsolve is a general CFD solver allowing multi-material, multi-phase and multi-species calculations to be considered. The general structure is illustrated below in 19.2.

Global Domain The global domain considers the entire computation and can consist of a number of fluid and solid parts.

Fluid Domain Each of the fluid domains represents a liquid or gas or a mixture of both within the calculation.

Fluid Phase Fluids may consist of single or multiple phases - which can be liquid or gases or a mixture of both.

Species/Components Each of the fluid phases may consist of a single or multiple species.

Solid Domain Each of the solid parts may consist of many different materials.

Material A solid domain may consist of a number of different materials, each of which can have different material properties.

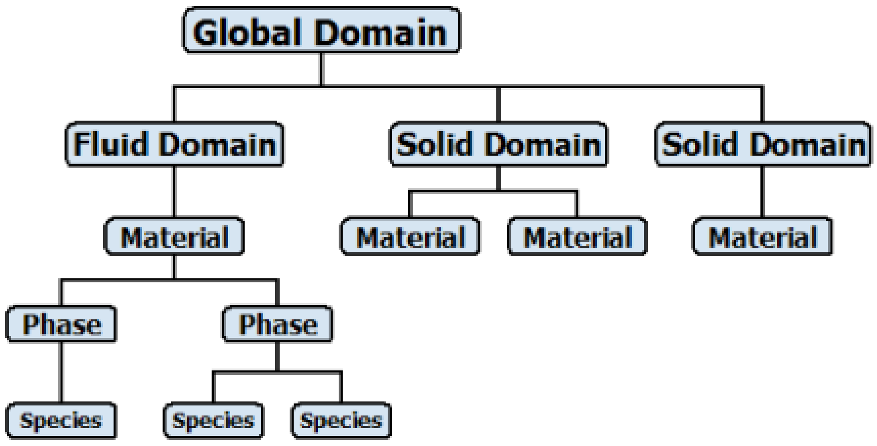
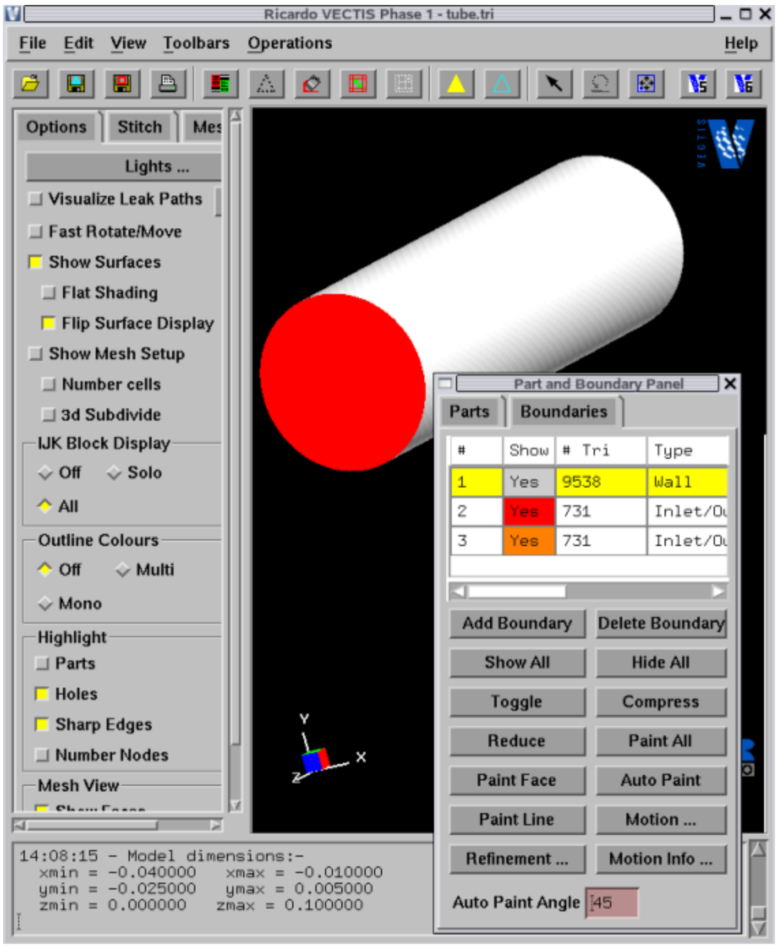


Figure 19.2: VECTIS Solver Structure

This first example shows how to set-up and run a basic flow analysis It will assume a basic familiarity with performing CFD calculations.

19.1.3 Geometry Preparation

Phase 1 is the current pre-processing package for use with VECTIS. Phase 1 is used to read in geometry and process it to a form acceptable to the VECTIS mesh generator. Depending on the quality of the initial geometry a certain amount of repair may be needed to form a single closed volume. Once this is complete Phase 1 is then used to identify regions of the geometry as different boundaries for the CFD calculation. Finally the 'global mesh' and other control parameters for the mesher are defined. Further information can be found in the chapter GEOMETRY.



#	Show	# Tri	Type
1	Yes	9538	Wall
2	Yes	731	Inlet/Ou
3	Yes	731	Inlet/Ou

```
14:08:15 - Model dimensions:-
  xmin = -0.040000   xmax = -0.010000
  ymin = -0.025000   ymax = 0.005000
  zmin = 0.000000   zmax = 0.100000
```

Figure 19.3: Geometry Preparation and Boundary Painting

In this tutorial case the above is achieved in the following steps.

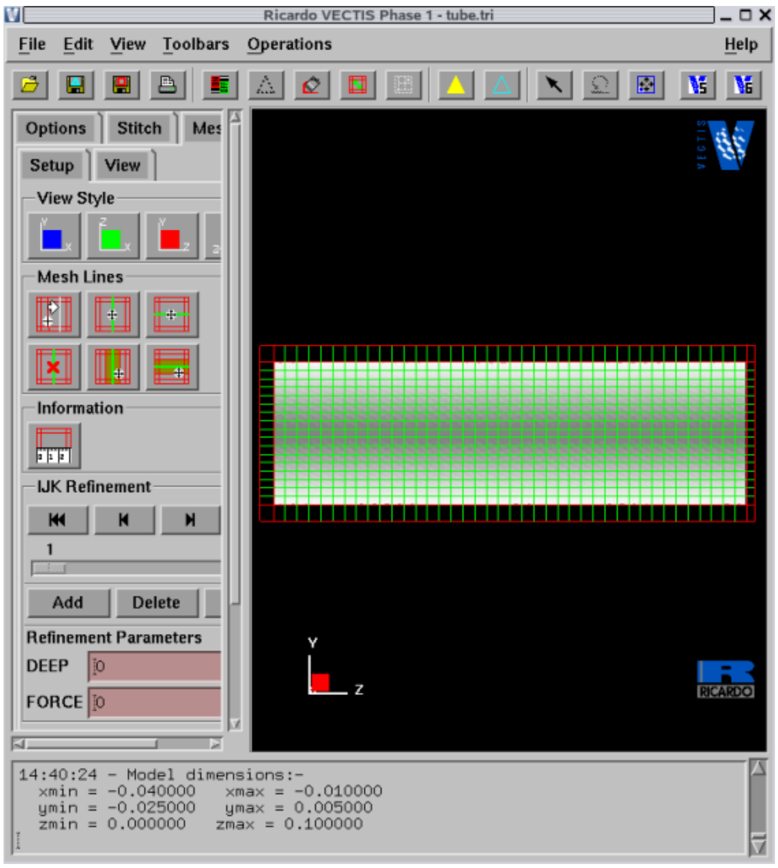
1. Open Phase1 (either typing phase1 in a command prompt or from the start menu)
2. Open the file tube.tri (*'File'* > *'Open'* then select 'tube.tri')
3. Open the boundary painting panel (*'operations'* > *'boundary painting'*)
4. Add two new boundaries
5. Paint each end of the tube as a separate boundary (using paint face)

6. Change the two ends to inlet/outlet boundaries (click on type in boundary panel)

The prepared geometry should appear as in Fig 19.3

19.1.4 Defining the Global Mesh

The purpose of this section of Phase 1 is to set up the global control mesh lines used as the basis for mesh generation. The mesh setup tools are activated by selecting the *'Toolbars'* > *'Mesh Setup'* option in the Phase1 menu. The tools are displayed and explained in Fig 19.4. A default outline mesh is displayed, which can be manipulated to provide the required control mesh. Here a global mesh size of approximately 2.5 mm is used. It is important to remember that the mesh needs to be defined in all three directions; so at least two of the x-y, x-z, y-z views need to be used. Once the global mesh is defined save the mesh (*'File'* > *'Save mesh as'*) as 'tube.mesh'

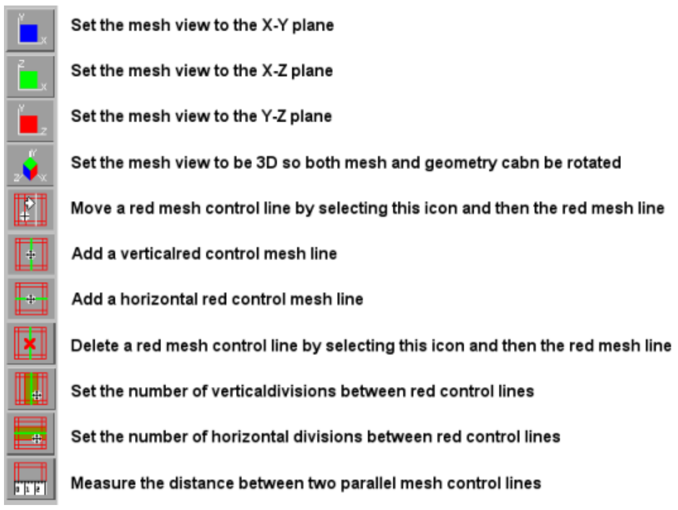


```
14:40:24 - Model dimensions:-
  xmin = -0.040000   xmax = -0.010000
  ymin = -0.025000   ymax = 0.005000
  zmin = 0.000000   zmax = 0.100000
```

Figure 19.4: Global Mesh Definition

19.1.5 R-Desk

R-Desk GUI



Set the mesh view to the X-Y plane

Set the mesh view to the X-Z plane

Set the mesh view to the Y-Z plane

Set the mesh view to be 3D so both mesh and geometry cabn be rotated

Move a red mesh control line by selecting this icon and then the red mesh line

Add a verticalred control mesh line

Add a horizontal red control mesh line

Delete a red mesh control line by selecting this icon and then the red mesh line

Set the number of verticaldivisions between red control lines

Set the number of horizontal divisions between red control lines

Measure the distance between two parallel mesh control lines

Figure 19.5: Options to manipulate mesh lines

Open R-Desk either by typing ’rdesk’ at the command prompt or using the start menu. Create a new VECTIS sesion through the menu using *’Menu’* > *’New’* .

The default panel layout for a VECTIS project will be shown, however panels can be re-arranged by dragging and docking them into the desired format. Additional panels can be opened by right clicking the project menu bar (Fig. 19.6). Once a desired layout is found it can be saved and also be made as the default using *’File’* > *’Layout’* > *’Save’*.

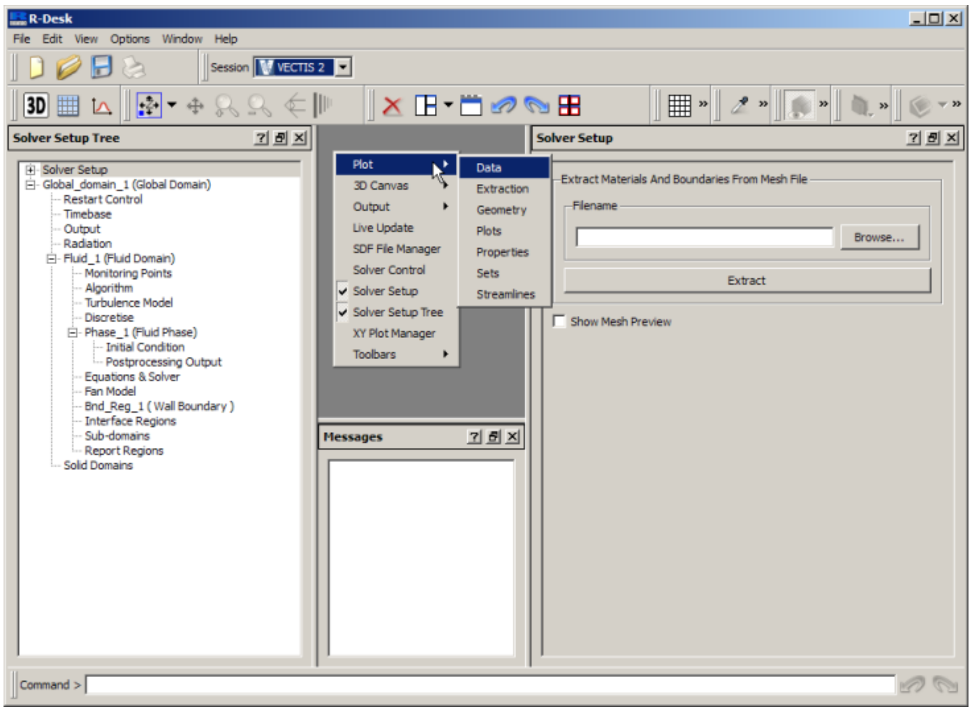


Figure 19.6: R-Desk Layout

19.1.6 Generating the Grid File

Once the global mesh has been defined the mesh generator, vmesh, can be started using Launch Mesher button in R-Desk (Fig 19.7). Once the panel has opened, browse to the 'tube.mesh' file saved from Phase1. Then click launch

Alternatively the mesher can be started from a command window. Type *'vmesh tube.mesh'* at the prompt

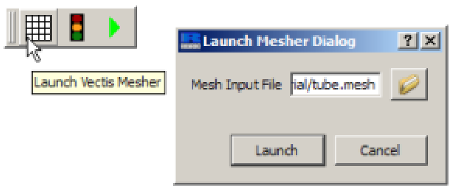


Figure 19.7: Launching the mesh generator in R-Desk

This will write output to the command window.

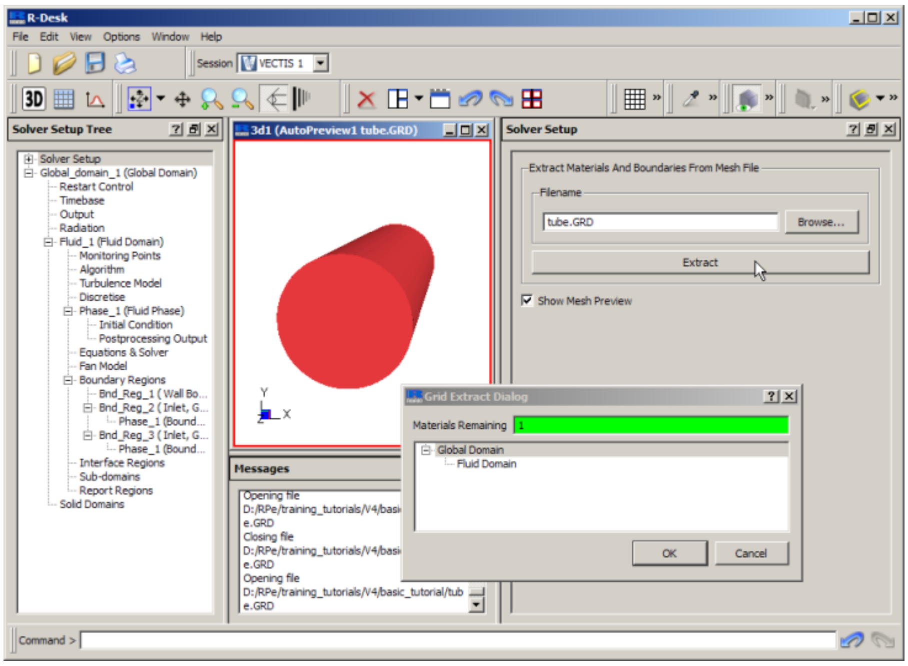
```
..........( 50 % done )
..........( 60 % done )
..........( 70 % done )
..........( 80 % done )
..........( 90 % done )
..........(100 % done )
- all common faces successfully generated

-----------------------------------------------------------------
STATISTICAL DATA OF GENERATED MESH
Number of generated cells: 14461 (boundary: 5281 ; internal: 9180)
--- Patching method ---
Number of cells processed by Marching Cubes: 4881 (92.43 %)
Number of cells processed by Exact Fit: 299 (5.66 %)
Number of volumes broken by Cell Splitting: 0 (0.00 % of boundary cells)
--- Cell quality ---
Number of correct boundary cells: 5180 (98.09 %)
NO PROBLEMS with negative volumes
NO PROBLEMS with gaps
There were 130 small volumes, they are deleted now (2.46 %)
Number of cells which had to be deactivated: 101 (1.91 %)
Total mesh volume: 7.12103e-005 u3
[ = 6.39150e-005 u3 (9180 inner cells) + 7.29531e-006 u3 (5180 boundary cells)]
-----------------------------------------------------------------


WRITING THE MESH FILE
- successfully finished

Total time elapsed: 4 seconds

SUCCESSFULLY DONE
```

A computational mesh is produced (tube.GRD)

19.1.7 Importing the Grid File

The materials and properties defined in a grid file can be imported into the solver setup. This will populate the solver setup tree with the correct number of materials and boundaries. A .GRD file needs to be entered into the 'Filename' box. Clicking on the browse button will open a dialog box and allow the relevant grid file to be selected. Once a file has been selected. The name will appear in the Filename box. Next, click the extract button.

A dialog box pops up and displays the materials found in the imported grid file. Then each material in the grid file needs to be allocated into a fluid or a material in a solid domain.



Opening file
D:/RPe/training_tutorials/V4/basi
e.GRD
Closing file
D:/RPe/training_tutorials/V4/basi
e.GRD
Opening file
D:/RPe/training_tutorials/V4/basic_tutorial/tub
e.GRD

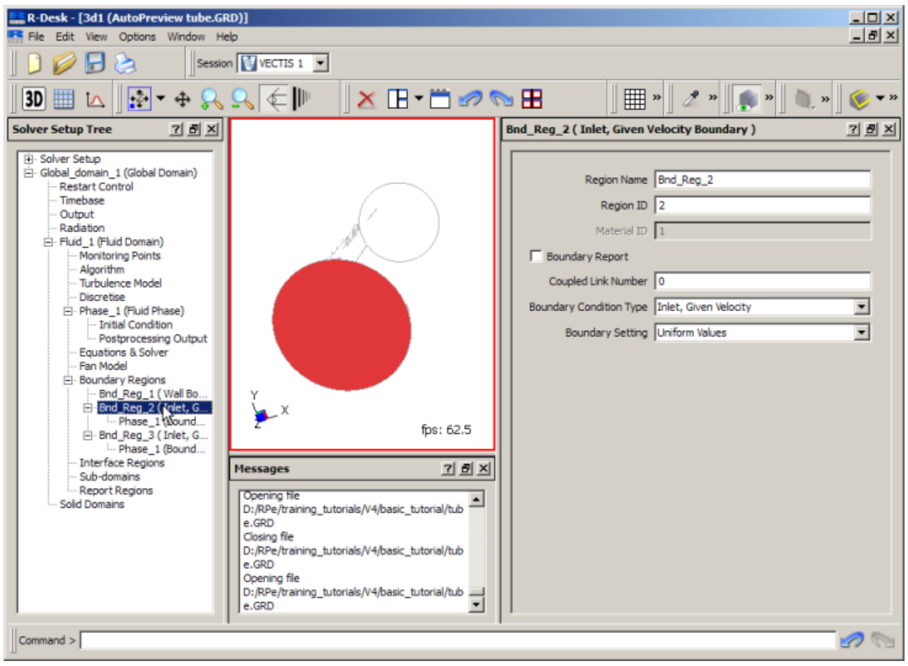
Figure 19.8: Importing the mesh into R-Desk Solver Setup

By importing the grid file the relevant regions of the grid are highlighted when they are selected in the solver setup tree . For example in 19.9, the first boundary region is selected in the tree and is shown in the preview window.

19.1.8 Solver Setup

The input to the solver is specified using the solver setup tree and solver setup input panels. The structure of the tree reflects the general structure of calculation.

- Global domain containing general options for the calculation, timebase etc..
- Fluid domains containing data for phases, species and boundaries
- Solid domains for materials and boundaries


R-Desk - [3d1 (AutoPreview tube.GRD)]
File Edit View Options Window Help
Session VECTIS 1
Solver Setup Tree
Bnd_Reg_2 (Inlet, Given Velocity Boundary)
Messages
fps: 62.5
Command >

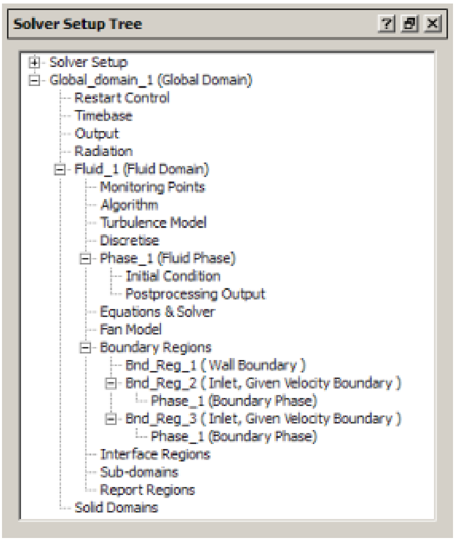
```
Solver Setup
Global_domain_1 (Global Domain)
    Restart Control
    Timebase
    Output
    Radiation
    Fluid_1 (Fluid Domain)
        Monitoring Points
        Algorithm
        Turbulence Model
        Discretise
        Phase_1 (Fluid Phase)
            Initial Condition
            Postprocessing Output
        Equations & Solver
        Fan Model
        Boundary Regions
            Bnd_Reg_1 ( Wall Bo...
            Bnd_Reg_2 ( Inlet, G...
                Phase_1 (Bound...
            Bnd_Reg_3 ( Inlet, G...
                Phase_1 (Bound...
        Interface Regions
        Sub-domains
        Report Regions
    Solid Domains
```

Region Name	Bnd_Reg_2
Region ID	2
Material ID	1
Boundary Report	☐
Coupled Link Number	0
Boundary Condition Type	Inlet, Given Velocity
Boundary Setting	Uniform Values

```
Opening file
D:/RPe/training_tutorials/V4/basic_tutorial/tub
e.GRD
Closing file
D:/RPe/training_tutorials/V4/basic_tutorial/tub
e.GRD
Opening file
D:/RPe/training_tutorials/V4/basic_tutorial/tub
e.GRD
```

Figure 19.9: Selecting a boundary in the solver setup input tree

The contents of the in solver setup input panel will change corresponding to the selected entry in the solver setup tree .



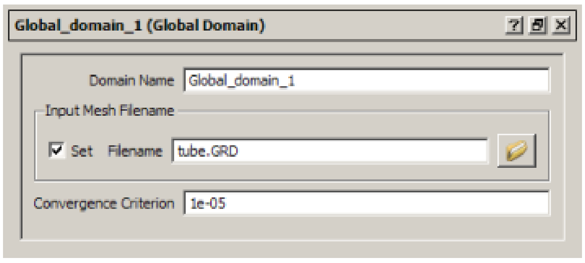
```
Solver Setup
Global_domain_1 (Global Domain)
    Restart Control
    Timebase
    Output
    Radiation
    Fluid_1 (Fluid Domain)
        Monitoring Points
        Algorithm
        Turbulence Model
        Discretise
        Phase_1 (Fluid Phase)
            Initial Condition
            Postprocessing Output
        Equations & Solver
        Fan Model
        Boundary Regions
            Bnd_Reg_1 ( Wall Boundary )
            Bnd_Reg_2 ( Inlet, Given Velocity Boundary )
                Phase_1 (Boundary Phase)
            Bnd_Reg_3 ( Inlet, Given Velocity Boundary )
                Phase_1 (Boundary Phase)
        Interface Regions
        Sub-domains
        Report Regions
    Solid Domains
```

Figure 19.10: The solver setup input tree

19.1.8.1 Solution Control

The Solution control panel allows general simulation parameters to be set.

Domain Name A name can be given to each global domain in the calculation. For example 'Coolant' or 'Cylinder Head'.


Global_domain_1 (Global Domain)

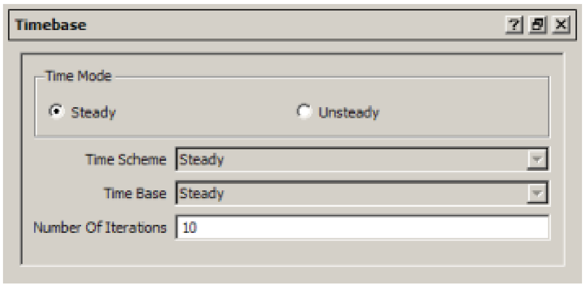
Domain Name: Global_domain_1

Input Mesh Filename: [✓ Set] Filename: tube.GRD

Convergence Criterion: 1e-05

Figure 19.11: The Global Domain Panel

Input Mesh Filename The solver will default to using a computational grid file with the name projectname.GRD. A different grid file can be specified here. (otherwise the projectname is taken as the .inp file prefix)


Timebase

Time Mode: [✓ Steady] Unsteady

Time Scheme: Steady

Time Base: Steady

Number Of Iterations: 10

Figure 19.12: The Timebase Panel

Time Mode Here the type of calculation time mode is selected

1. Steady
   - A maximum number of iterations is chosen for the calculation.
   - If all the residuals for all the equations are less than the specified tolerance the calculation will terminate.
2. Unsteady

Time Scheme This can be set to either '1st Order Euler' or '2nd Order Implicit'

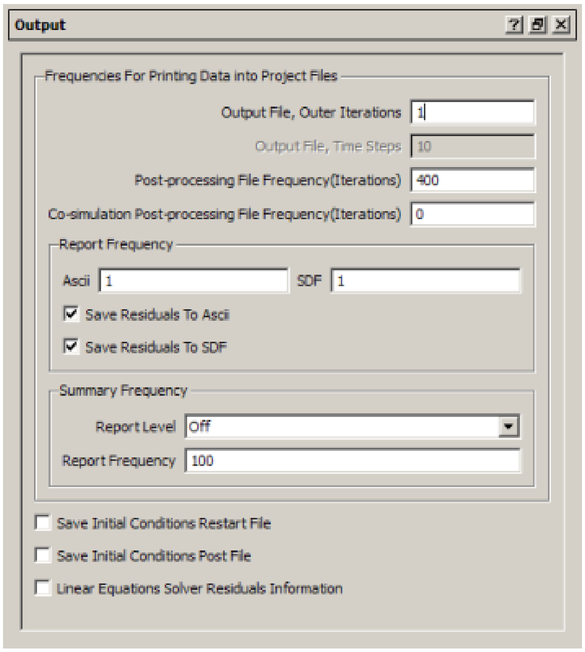
Time Base Here the time base is selected

- 2-stroke
- Seconds
- 4-stroke
- Non-Dimensional

Time Dependant Boundary Conditions The boundary conditions can be steady or unsteady/transient, which is determined by selecting the 'time dependent boundary conditions' toggle.

Timesteps The number of timesteps is chosen for the calculation, the end time being the the chosen time step multiplied by the number of timesteps specified. Additionally, a 'Maximum Number of Outer Iterations' is chosen. This is the maximum number of iterations performed per time step.

– Typically the number of iterations required per step, will reduce after the initial period of the calculations. However this will be dependent on the accuracy of the applied initial conditions.



Output

Frequencies For Printing Data into Project Files

Output File, Outer Iterations: 1

Output File, Time Steps: 10

Post-processing File Frequency(Iterations): 400

Co-simulation Post-processing File Frequency(Iterations): 0

Report Frequency

Ascii: 1 SDF: 1

☑ Save Residuals To Ascii

☑ Save Residuals To SDF

Summary Frequency

Report Level: Off

Report Frequency: 100

☐ Save Initial Conditions Restart File

☐ Save Initial Conditions Post File

☐ Linear Equations Solver Residuals Information

Figure 19.13: The Output Panel

Reporting

General output Here the frequency of data reporting output is chosen. Firstly the monitoring and convergence data written to the screen is also written to the .out file at the intervals specified. If zero values are used then the output is not written. The frequency of post-processing file writing is specified in the 'Post-processing File Frequency' box.

Additional output Additional output file can be written for monitoring data, boundary data, report regions and domain data. These can be either in ASCII column format or Ricardo SDF binary format. The frequency of the two types of file can be chosen independently by entering the required intervals in to the 'Report Frequency' boxes. Again an interval of zero will mean that the data is not written. The ASCII files include header rows detailing the data in each of the columns. Separate files are written for the different data types.

– Domain

– Monitoring

– IO and Wall (If reporting is selected for the relevant boundary)

– Arbitrary Surface

Residual data can also be written to and ascii file and to the report file by selecting the relevant check boxes.

The ASCII data files are used by the 'Live Update' utility in R-Desk. The SDF binary format is written to a single report file named projectname.rep_runnumber. It contains the data for all the additional output. It can be viewed and plotted in R-Desk using the 'SDF File Manager' and 'XY Plot Manager'

Summary output Summary data can also be written to the screen and output file.

Co-Simulation Post-processing Frequency Controls the frequency at which post-processing data is written when WAVE requests postprocessing output during a coupled simulation.

Save Initial Conditions The 'Save Initial Conditions Restart File' and 'Save Initial Conditions Post File' toggles allow the user to select whether the initial data used in the calculations is saved to a file. The initial conditions are saved after any flow potential calculations are performed but before the first timestep/iteration.

Linear Equations Solver Residuals Information This toggles whether residual information is written for each of the equations at every timestep. This will write to the screen and output files in the format shown below.

This pipe flow example uses the following main options.

- Steady state
- 1000 iterations
- Convergence criterion 1e-5

During the calculation, restart files are written at a user defined frequency throughout the calculation specified in the 'Restart File Frequency' field. Two restart files are generated and are written alternately at the restart frequency time step interval.

Restarting the calculation To restart the calculation from a saved restart file, the 'Restart With Previous Results' toggle must be selected. By default this will use the most recent restart file written for the current project. Additionally a filename can be explicitly specified to restart the calculation from an alternate .rst file. Once the 'Set Filename' Toggle is selected a restart filename can be entered into the 'Filename' box. Timed restarts can also be specified to allow the writing of restart files at specific times, with specific names. A time value (in terms of iteration or timestep depending on whether the calculation is steady or unsteady) and a filename to define a restart specification.

This pipe flow example uses the default options

19.1.8.2 Fluid Domain

In this case the calculation will consist of one single phase fluid domain

Domain Name A name can be assigned to the fluid domain to allow for easier reference later in the calculation.

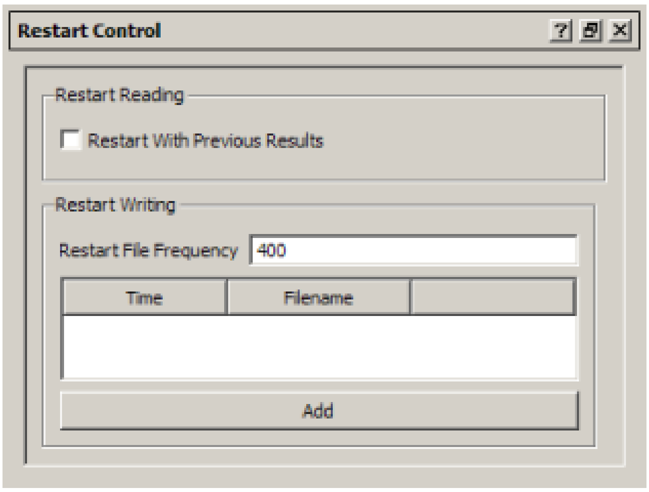


Figure 19.14: The Restart Panel

Material Name A name can be assigned to each material in the fluid domain, again to allow for easier reference later in the calculation. Each fluid domain contains only one material

Material ID This refers to the material ID number referenced in the GRD file. Each separate material will have a unique ID number

Multiphase Modelling This allows the user to specify whether the fluid contains a single phase or a homogeneous mixture of a number of different phases.

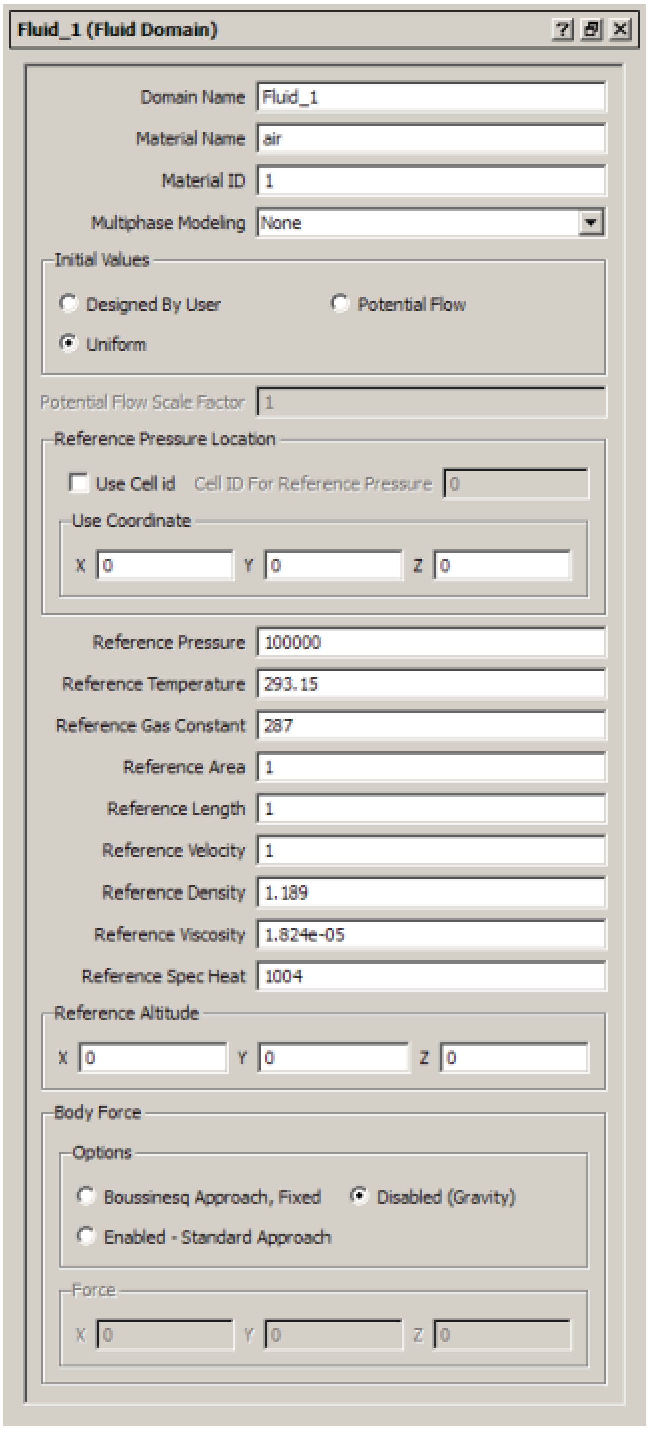
Initial Values Initial velocity, pressure, temperature, passive scalar, turbulence and species mass fraction for the solution domain at the start time, as well as providing some other initial conditions options.

- Designed by User (This option allows initial conditions to be specified using a user routine.)
- Potential Flow (This option will calculate the initial flow field based on the boundary condition data specified. The calculated velocity values can then be scaled using the 'Potential Flow Scale Factor'. This is useful because the calculated potential flow may be much higher than the actually flow as turbulence and viscous effects are not taken into account.
- Uniform

Monitoring Monitoring points are specified within the model domain where values of the solution are output at the end of each time step.

- Pressure Monitoring - A cell ID number or X,Y,Z co-ordinate can be specified for the reference pressure values.
- Velocity Monitoring - A cell ID number or X,Y,Z co-ordinate can be specified for monitoring velocity values.

These values are written to the screen and to the output file.


Fluid_1 (Fluid Domain)

Domain Name: Fluid_1
Material Name: air
Material ID: 1
Multiphase Modeling: None

Initial Values
( ) Designed By User ( ) Potential Flow
(●) Uniform

Potential Flow Scale Factor: 1

Reference Pressure Location
[ ] Use Cell id Cell ID For Reference Pressure: 0
Use Coordinate
X: 0 Y: 0 Z: 0

Reference Pressure: 100000
Reference Temperature: 293.15
Reference Gas Constant: 287
Reference Area: 1
Reference Length: 1
Reference Velocity: 1
Reference Density: 1.189
Reference Viscosity: 1.824e-05
Reference Spec Heat: 1004

Reference Altitude
X: 0 Y: 0 Z: 0

Body Force
Options
( ) Boussinesq Approach, Fixed (●) Disabled (Gravity)
( ) Enabled - Standard Approach
Force
X: 0 Y: 0 Z: 0

Figure 19.15: The Fluid Domain Panel

Reference Data The reference values should have representative values for the solution.

Body Force Options to allow the simulation of body force.

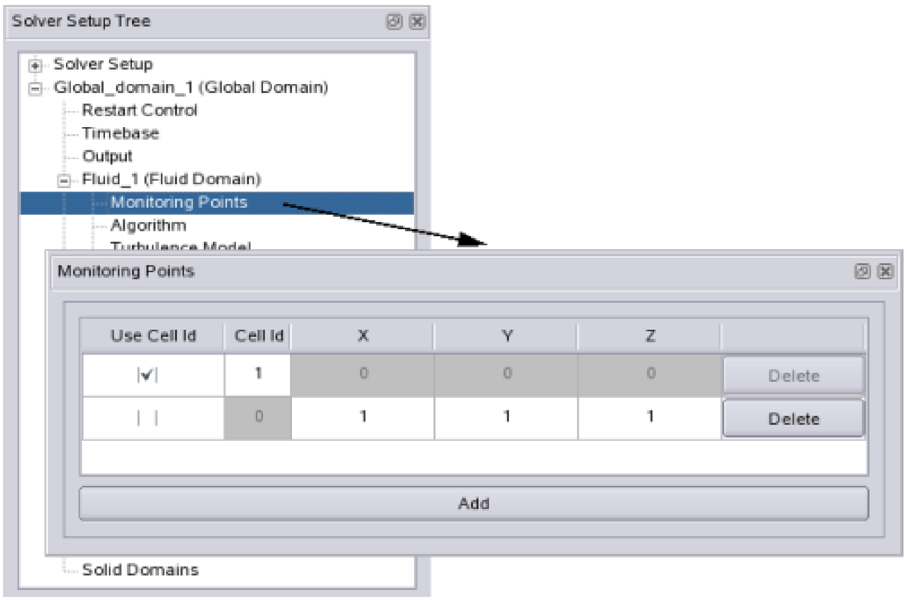
Here the pipe flow example uses water as the fluid, the following reference values are used.

– Density - 1012

– Viscosity - 0.000719

– Specific heat - 3620

For the moment the the initial values will be set as uniform. For all the other data entries on the panel the default values will be used. Additionally for this example the remaining fluid panels (algorithm, turbulence model, equations & solver and discretise) will be left at their default values.



Use Cell Id	Cell Id	X	Y	Z	
✓	1	0	0	0	Delete
	0	1	1	1	Delete

Figure 19.16: The Solver Setup Tree

Next the fluid phase needs to be set-up.

19.1.8.3 Fluid Phase

The fluid phase panel contains data for each fluid phase. Each phase may be made up of a single or multiple species.

Phase Name Each phase can be given a name to aid reference.

Mixture of Species Option Each phase can be made up of a number of different components/species. In this tutorial a single-component phase is considered. More details of the different options can be found in 'Setting a multiphase mixture'.

Phase Type and compressibility The phase type can be selected to be either a gas or a liquid. In the case of liquid simulation the compressibility is set to be incompressible. For gases there is a choice of compressibility options;

– Incompressible Fluid,

– Weakly Compressible

– Fully Compressible, Subsonic

  - Fully Compressible, Supersonic

  For the tutorial we specify an incompressible liquid. More details of the different options can be found in 'Setting a phase and its properties'.

Solve all species This determines whether the volume fraction equations for all phases is solved or whether the last phase is taken as the remaining volume fraction after the other phases are solved.

Property Specification In general there are a number of ways of specifying the properties; constant values or relationships.

In this case the values for water are fixed as constants.

- Density - 1012
- Viscosity - 0.000719
- Specific heat - 3620

Initial Conditions

The initial conditions for the fluid can be specified in the initial conditions panel. This allows the user to specify initial flow velocity, pressure, temperature etc...

These values will be imposed on the fluid when uniform initial conditions are selected in the Fluid Domain panel

Output

The data to be output to the post file can be selected in the output panel. Each of the scalar variable checked will be written to the file.

19.1.8.4 Boundary Regions

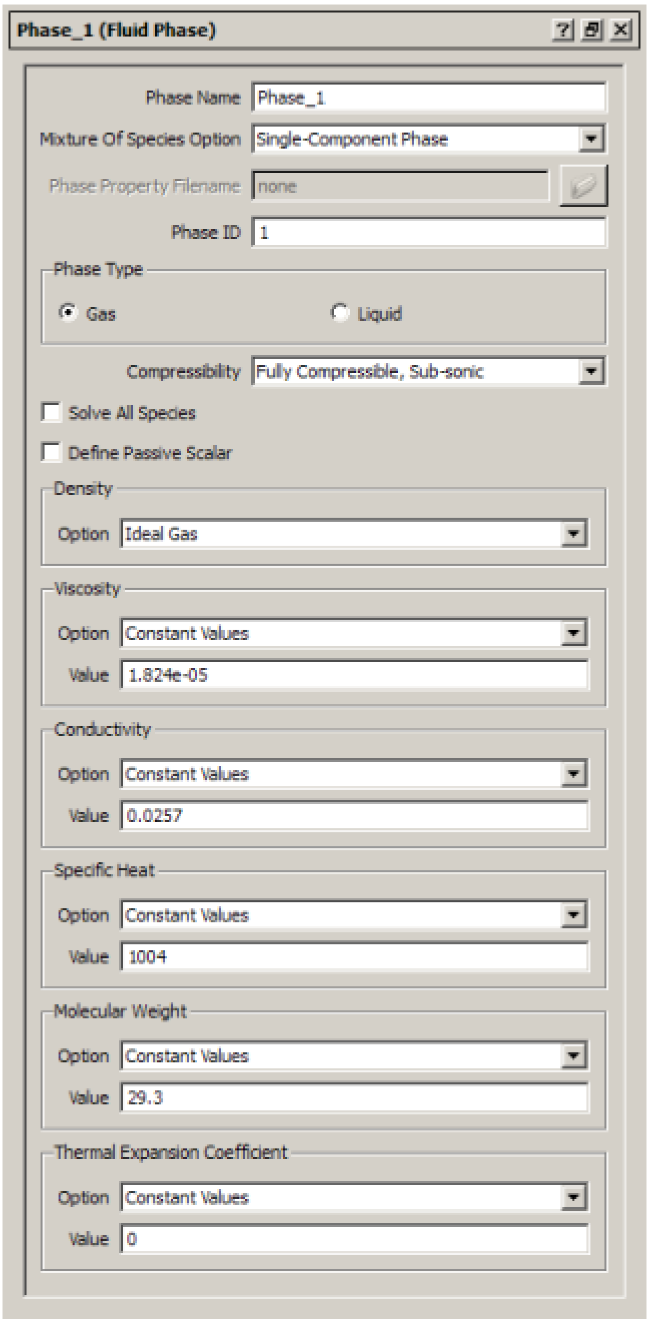
Each of the boundary regions can then be specified.

Region Name A name can be given to each boundary for reference. The name can be referenced when using user programming routines.

Boundary report This specifies whether output data is written to the report files for this boundary

Coupled Link Number When running coupled WAVE/VECTIS the interface numbers for each coupled boundary are specified here.

Boundary Condition Type Next the appropriate boundary type is chosen for each boundary


Phase_1 (Fluid Phase)

Phase Name	Phase_1
Mixture Of Species Option	Single-Component Phase
Phase Property Filename	none
Phase ID	1

Phase Type: (● Gas) ○ Liquid

Compressibility: Fully Compressible, Sub-sonic

☐ Solve All Species

☐ Define Passive Scalar

Density — Option: Ideal Gas

Viscosity — Option: Constant Values; Value: 1.824e-05

Conductivity — Option: Constant Values; Value: 0.0257

Specific Heat — Option: Constant Values; Value: 1004

Molecular Weight — Option: Constant Values; Value: 29.3

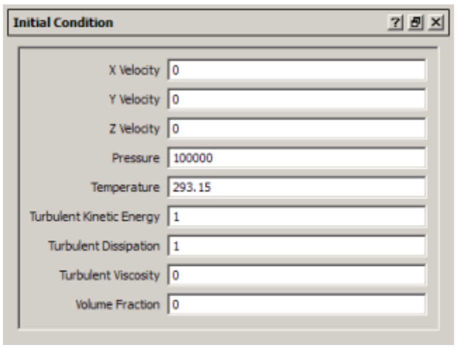
Thermal Expansion Coefficient — Option: Constant Values; Value: 0

Figure 19.17: The Fluid Phase Panel

Details of the different boundary conditions can be found in the BOUNDARY CONDITION TYPES chapter.

Here we will set

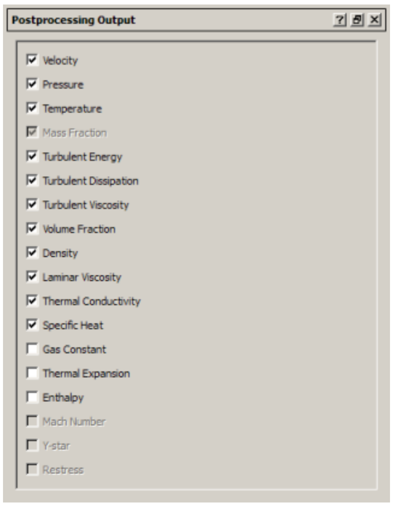
- Boundary 1 to be a wall with a fixed temperature of 450 degK
- Boundary 2 to a mass-flow boundary with a flow in of 0.1kg/s, an initial pressure of 1.001 bar and temperature of 300 degK
- Boundary 3 will be a static pressure boundary set to 1 bar, outflow and with a temperature of



Initial Condition

X Velocity	0
Y Velocity	0
Z Velocity	0
Pressure	100000
Temperature	293.15
Turbulent Kinetic Energy	1
Turbulent Dissipation	1
Turbulent Viscosity	0
Volume Fraction	0

Figure 19.18: Initial Conditions



Postprocessing Output

- ☑ Velocity
- ☑ Pressure
- ☑ Temperature
- ☑ Mass Fraction
- ☑ Turbulent Energy
- ☑ Turbulent Dissipation
- ☑ Turbulent Viscosity
- ☑ Volume Fraction
- ☑ Density
- ☑ Laminar Viscosity
- ☑ Thermal Conductivity
- ☑ Specific Heat
- ☐ Gas Constant
- ☐ Thermal Expansion
- ☐ Enthalpy
- ☐ Mach Number
- ☐ Y-star
- ☐ Restress

Figure 19.19: The Fluid Phase Output Panel

300 degK

The input file should now be complete for the analysis Save the file as tube.inp (’file’ > ’save as’)

19.1.9 Grid Preparation

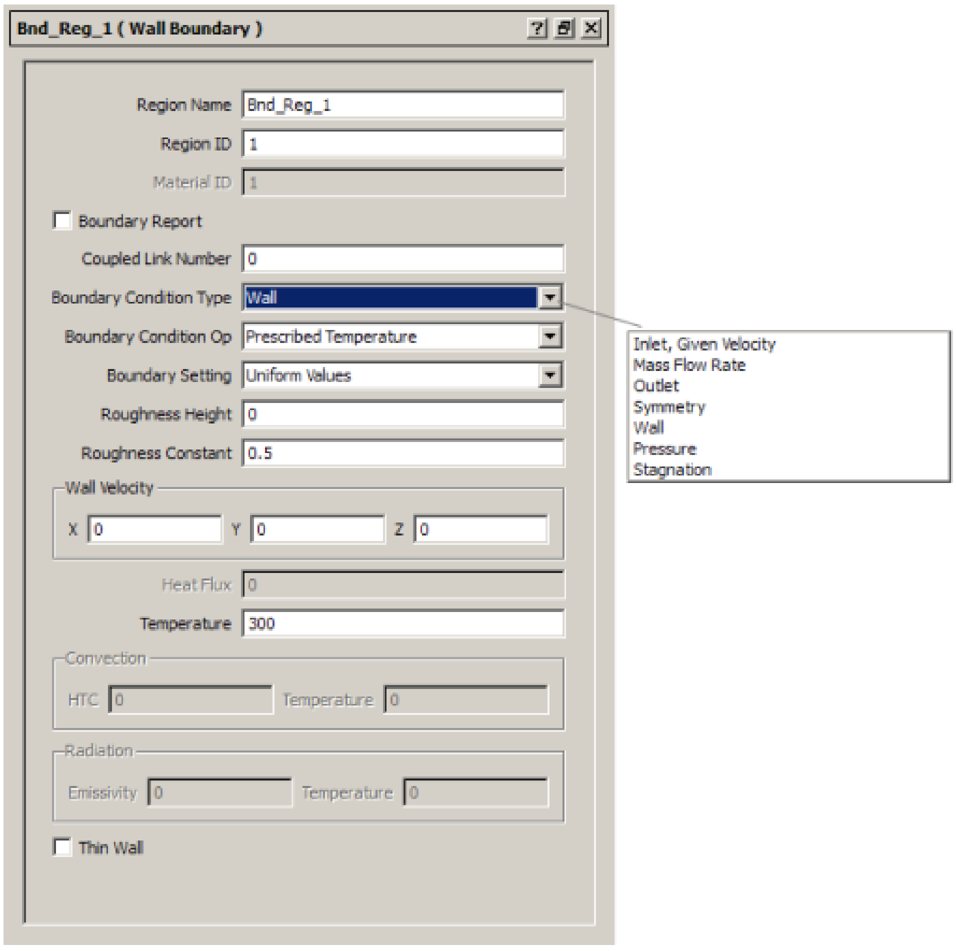
In a command window run the *vpre* mesh pre-processor. This will re-order the grid for use with the solver. *vpre* can also be used to re-partition the mesh for parallel calculations.

Type ’vpre tube.GRD’ in the cmd window

19.1.10 Running the solver

Running the solver

The simulation is now ready to be run. Type ’vsolve tube.GRD’ in a command window The solver will start


Bnd_Reg_1 (Wall Boundary)

Region Name	Bnd_Reg_1
Region ID	1
Material ID	1
☐ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Wall
Boundary Condition Op	Prescribed Temperature
Boundary Setting	Uniform Values
Roughness Height	0
Roughness Constant	0.5

Wall Velocity: X 0, Y 0, Z 0

Heat Flux	0
Temperature	300

Convection: HTC 0, Temperature 0

Radiation: Emissivity 0, Temperature 0

☐ Thin Wall

Figure 19.20: The Boundary Region Panel

19.1.11 Live Update

Live update is a utility in R-Desk that allows a simulation to be monitored whilst it is running. Firstly open a xy canvas using the new XY canvas button. Then in the Live Update panel, browse to the directory where the simulation is running. The available data files are presented in the 'Files' window. The data files available are determined by the ascii files selected in the reporting section of the 19.1.8.1Global Domain in the solver setup tree. Once a file is selected the data available to be plotted is shown in the 'Value' window. The different data can be dragged onto a xy plot canvas The xy plot can then be further modified using the XY plot manager

19.1.12 Post-processing

Once the simulation converges it will stop and write restart and post-processing files. Open a 3D canvas. Open the post file in R-Desk using the *'File'* > *'Open'* > *'File'* menu option. It should appear in the plot tree. The plots from the tree can be dragged onto a 3D canvas

The attributes of the plot can be modified

Different data can be shown on the plot by selecting it in the data panel

19.1.12.1 2D plane

A 2D slice can be made of a plot Right-click the plot in the tree and select slice The slice is then defined in the slice definition panel

The slice can be previewed on the plot if it is present in a 3D canvas

Once the slice is in the correct position, click 'apply'

Open an additional 3D canvas These can be tiled using the 'window' > 'Tile... ' options Drag the slice plot into the second canvas

The data plotted on the slice can be changed using the data panel

19.1.13 Potential Flow Solver

We will now examine the potential flow solver option This option will calculate the initial flow field based on the boundary condition data specified.

Open the tube.inp input file The mesh file name needs to be specified in the solution control panel Also select 'Save Initial Conditions Post File' in this panel.

Next switch on the potential flow option for the initial values in the Fluid Domain data panel. Note: When potential flow solver is used to initialise the calculation it is important that the reference values and initial condition values specified are representative as they are used by the potential flow solver.

Save the file as tube_pot.inp ('file' > 'save as' ...) Run the simulation again (type 'vsolve tube_pot.inp' at the command prompt

Using live update the convergence of the two initialisation methods can be compared.

Using the potential flow initialisation has made the simulation converge much more rapidly when compared to the uniform constant initial conditions. This can be seen by the fact that the curves for the boundary mass flows when using the potential flow initialisation are much shorter (see Figure 19.21).

The initial flow field predicted by the potential solver can be examined by opening the tube_pot.post file. The step panel allows the iteration for which the data is plotted to be selected for each plot.

In this case a slice is made though the plot. The slice is then copied (right-click then 'copy') The first copy is placed in one canvas and iteration 'zero' is chosen, this shows the flow field after initialisation The second copy is dragged into another 3D canvas and data from the last step is selected in the step panel.

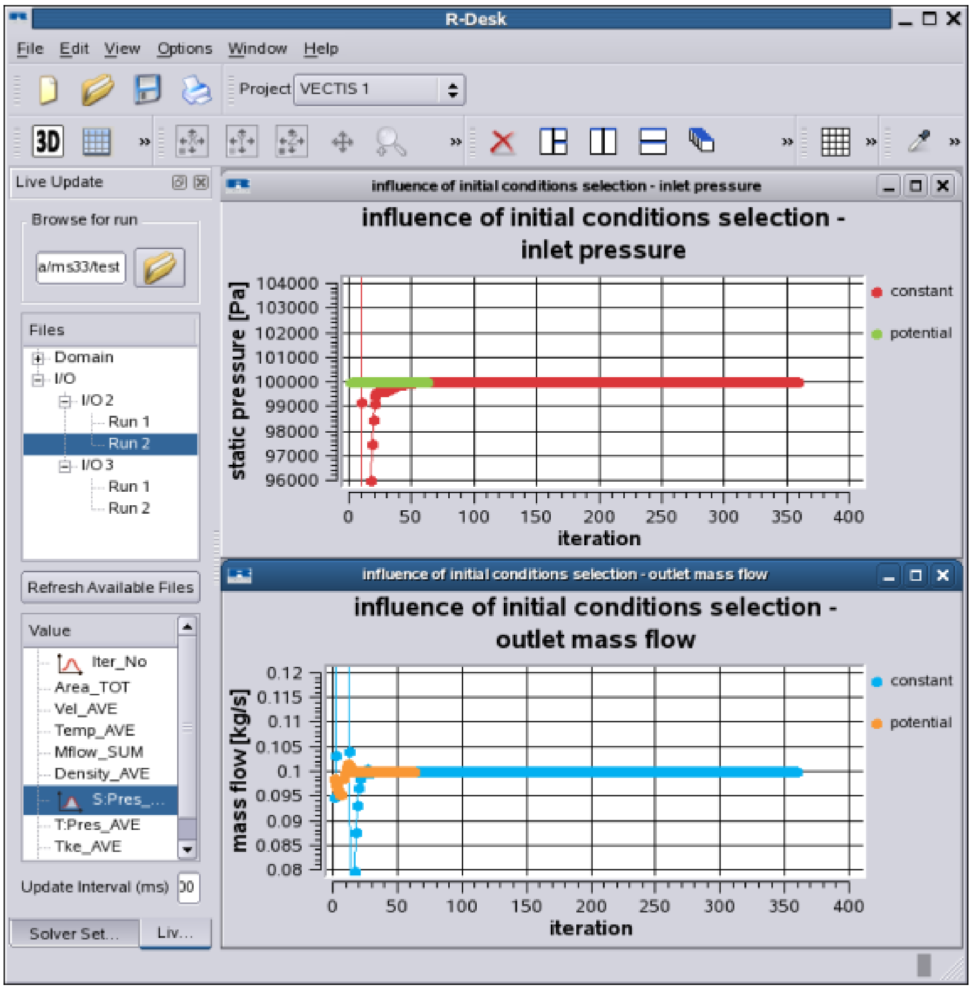


Figure 19.21: Comparison of IO mass flow convergence with different flow initialisation

19.2 Steady State Port Flow

Get the necessary files for the port flow tutorial

`http://www.software.ricardo.com/support/tutorials/vectismax/TutorialFiles`
`SSPortFlow`

19.2.1 Introduction

The purpose of this tutorial is to illustrate how to set-up and solve a steady state intake port flow calculation. The geometry represents a test rig and has a plenum chamber, intake port and valve.

19.2.2 Aims

Upon completion of this tutorial, the user should be able to perform the following:

- □ Define IJK mesh refinement blocks and use boundary refinement
- □ Activate the Steady State solver.
- □ Add Monitoring Points.
- □ Use arbitrary surfaces

The user should also be aware of boundary identification and the input parameters used for the solution of the simulation.

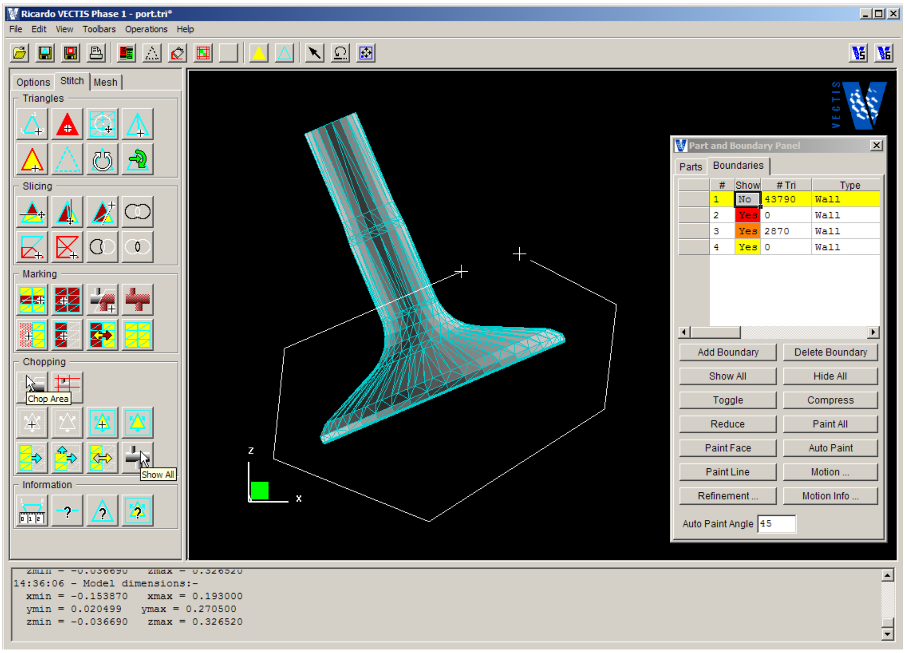
19.2.3 Geometry Preparation

Phase 1 is the current pre-processing package for use with VECTIS-MAX. Phase 1 is used to read in geometry and process it to a form acceptable to the VECTIS-MAX mesh generator. Depending on the quality of the initial geometry a certain amount of repair may be needed to form a single closed volume. Once this is complete Phase 1 is then used to identify regions of the geometry as different boundaries for the CFD calculation. Finally the 'global mesh' and other control parameters for the mesher are defined. Further information can be found in the GEOMETRY chapter.

In this tutorial the geometry is fully stitched so no further manual repair is necessary. The user can now enter the boundary identification section of Phase 1. The purpose of this section is to define different regions of the surface as different boundaries. In this example, the user needs to distinguish the different regions that will become the walls and inlet/outlet boundaries.

19.2.4 Selecting Boundaries

The boundaries can be selected either automatically or manually. For a geometry as simple as this it is recommended that the boundaries be defined manually. To do this use a combination of the 'Chop' and 'Mark Face' commands.



#	Show	# Tri	Type
1	No	43790	Wall
2	Yes	0	Wall
3	Yes	2870	Wall
4	Yes	0	Wall

Auto Paint Angle 45

```
  zmin = -0.036690   zmax = 0.326520
14:36:06 - Model dimensions:-
  xmin = -0.153870   xmax = 0.193000
  ymin = 0.020499   ymax = 0.270500
  zmin = -0.036690   zmax = 0.326520
```

Figure 19.22: Using the chop geometry command

The boundaries can be selected either manually or automatically. To manually select the boundaries, first open the Boundary Painting Tab in the Operations menu. Then select with the left mouse button the colour required by clicking on the number of the boundary required. Once a colour has been selected, the row that contains that colour will turn yellow. Then either the MARK LINE or a combination of the CHOP and MARK FACE commands can be used to select the different boundaries. The automatic method works on separating surfaces that are at angles greater to each other than is specified in the Auto Paint Angle option, the default being 45 degrees. Having specified an angle, selecting the Auto Paint button divides the various surfaces on the model into separate colours. The Reduce command can then be used to consolidate on the number of colours used and organise these so that the minimum number of colours are used, whilst still separating surfaces. Once the surfaces have been separated, the user may need to reorganise the sequence and/or contents of each colour so that items such as inlet/outlet boundaries are uniquely defined. For this example, the boundaries should be painted as described in Table 19.1. It is possible to drag and drop triangles from one boundary to another by clicking and dragging on the number of triangles in that boundary.

For this example, there are four boundaries - the inlet, defined as the back of the plenum chamber (i.e. opposite the inlet port); the outlet, defined as the bottom of the cylinder; the valve, used for boundary refinement; and the rest, which are walls.

19.2.5 Boundary Description

The figure below shows the definition of the inlet, outlet and main wall boundaries.

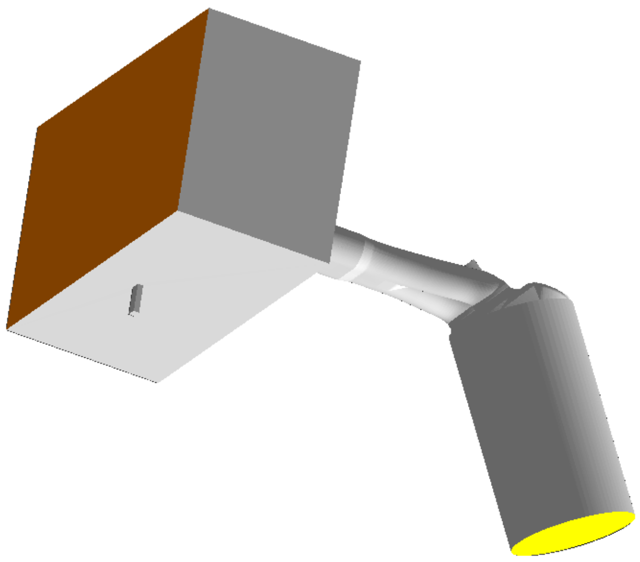


Figure 19.23: Inlet, outlet and main wall boundaries

Use the ’Chop’ geometry tool along with the ’Mark Face’ tool to define the back of the valves as a separate boundary, as shown in the figure below.

It is not strictly necessary to define the boundary types in Phase1 as they are later defined in the solver input file. However it is useful to define regions that are inlets and outlets as the mesher contains advanced options to allow different treatment depending on the boundary type.

Phase1 contains 4 possible boundary types that can be applied to the boundaries: Wall, Zero Gradient, Inlet/Outlet and Cyclic Symmetry. Only the Wall and Inlet/Outlet types have any significance in VECTIS-MAX and as stated above the solver input file allows these settings to be overridden.

Once the four boundaries have been defined the model should be saved as port.tri (’File’ > ’Save As’).

Boundary Description	Boundary Number
Wall	1
Wall (back of valve)	2
Inlet	3
Outlet	4

Table 19.1: Boundaries for Port Geometry

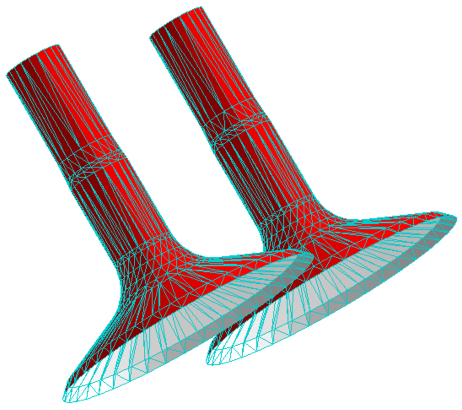
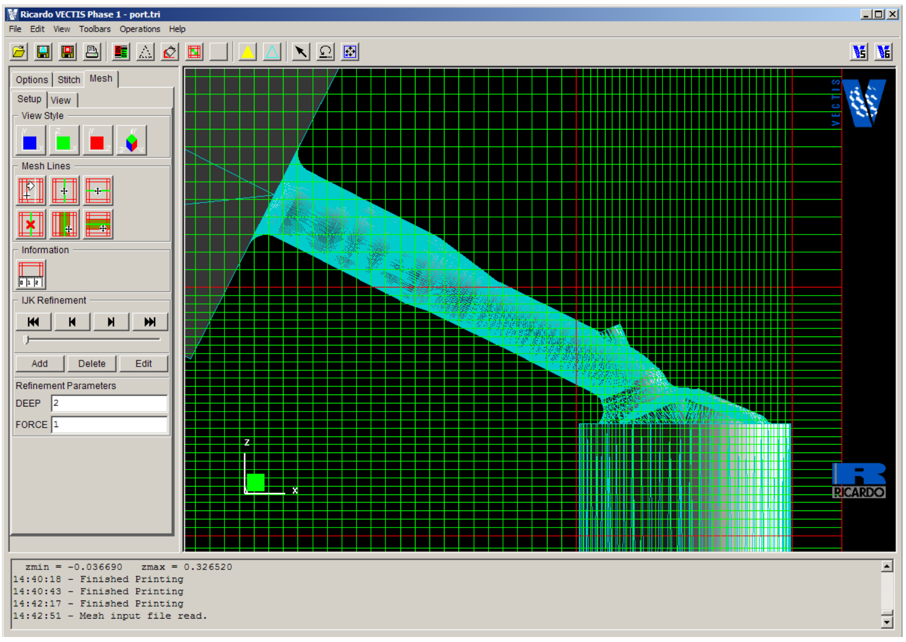


Figure 19.24: Back of valve defined as separate wall boundary to allow specification of boundary refinement

19.2.6 Mesh Specification

Having loaded a triangle file into phase1 select 'Toolbars' > 'Mesh Setup' from the menu. The purpose of this section of Phase 1 is to set up the global mesh lines which vmesh uses as the basis for mesh generation. A full description of the functionality of the Mesh Menu is described in the GEOMETRY chapter of the Documentation.

For this example, focus the mesh in the valve region using four or five major mesh lines. However, do not use too many cells. This is a training example and should run quickly. Use Figure 19.25 as a rough guide, it shows a global cell size of around 3mm in the valve region and 6mm in areas away from the valve. In addition to this, it is possible to use the IJK blocks to increase the detail level around the valve and inlet port.



Refinement Parameters
DEEP 2
FORCE 1

```
  zmin = -0.036690    zmax = 0.326520
14:40:18 - Finished Printing
14:40:43 - Finished Printing
14:42:17 - Finished Printing
14:42:51 - Mesh input file read.
```

Figure 19.25: Example of Global Mesh

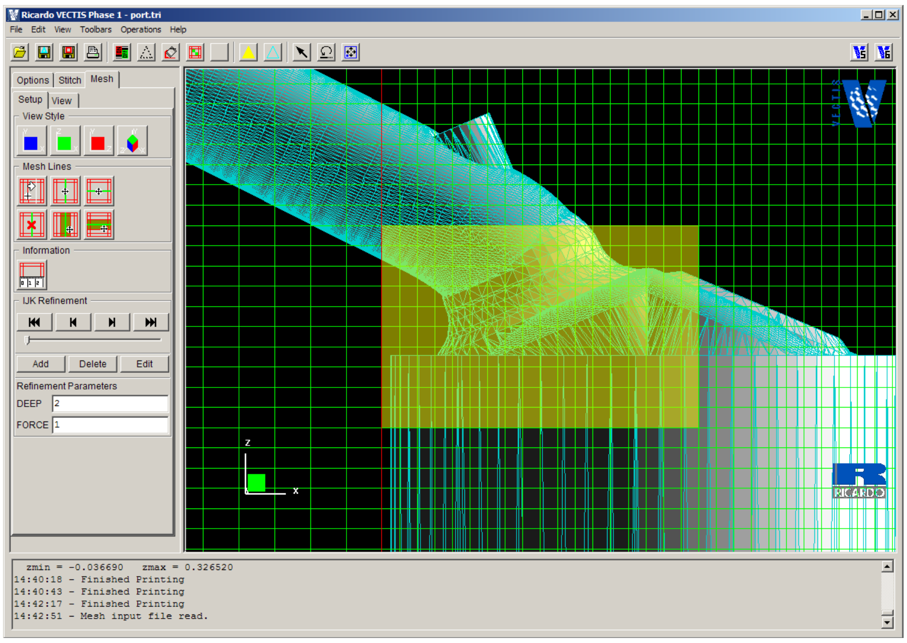
19.2.7 IJK Refinement

IJK refinement blocks are used to apply mesh refinement in specific regions of the geometry. The computational cells in the refined region are sub-divided according to the level of refinement applied.

In this case a single IJK block should be added to the geometry. The block should cover the valve region.

The IJK regions need to be defined in the three dimensions. However, as it would be difficult to define IJK blocks within the 3-D section of the mesh set-up menu, they must be defined two dimensions at a time.

In a 2-D Mesh View mode, the IJK Refinement Blocks panel as shown in Figure 19.26 will be used to add a refinement block. Firstly click the Add button, and draw out a box by pressing and holding the left mouse button to include the area required. If desired, this can be done using more than one IJK box - however, this is not strictly necessary. Having done this, change views to another 2D view using the Mesh View toolbar. Currently the IJK block defined in the previous view will be shown by a highlighted region and should be lined up along one edge of the mesh. Click Edit from the IJK Block Navigation window and draw another box by pressing and holding the left mouse button so as to include the cells required for the refinement region. Repeat for any other blocks. This should locate the IJK boxes in the correct location in 3D space. This can be checked by activating the 3D view from the Mesh View toolbar and turning on 3D sub-division and Show IJK block regions in the Viewing Options panel. The IJK block setup should be similar to that shown in Figure 19.26.



Refinement Parameters
DEEP 2
FORCE 1

```
  zmin = -0.036690   zmax = 0.326520
14:40:18 - Finished Printing
14:40:43 - Finished Printing
14:42:17 - Finished Printing
14:42:51 - Mesh input file read.
```

Figure 19.26: Setting IJK refinement

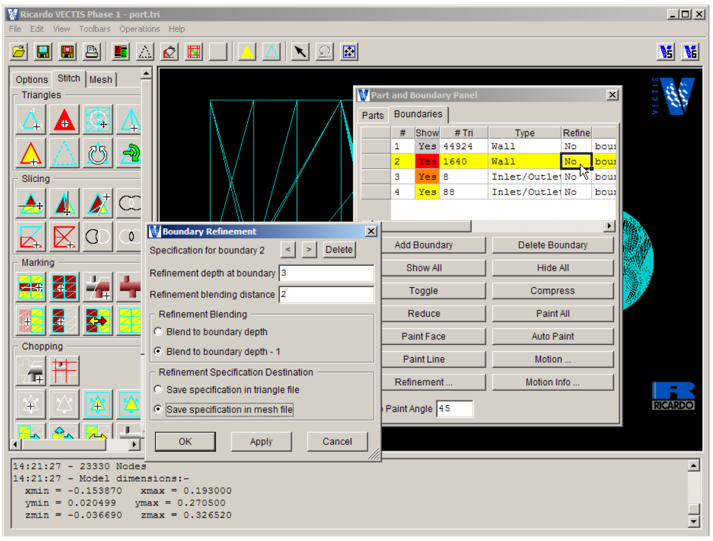
Next the required refinement depth for each IJK block must be set. Using the left and right buttons from the IJK Block Navigation window select the IJK block you require and set the DEEP and

FORCE refinement. These are set to 2 and 1 by default. They divide the mesh blocks within them into 2x2x2 or 4x4x4 blocks, for values of 1 and 2 respectively. Force values divide up all mesh blocks, whereas deep values only divide the mesh blocks that also have a boundary in them.

19.2.8 Boundary Refinement

Mesh refinement can additionally be specified on a per boundary region basis. This can useful when the region to be refined is complex or when refinement is used to capture particular geometric features.

Boundary refinement is specified in the 'Boundary Painting' panel used initially to define the separate boundary regions. Click on the refinement column for boundary 2, the boundary containing the back of the valves. Set the refinement level 'Refinement Depth at Boundary' to be 3. This means that cells in adjacent to the boundary will be sub-divided 3 times. Next set the 'Blending Distance' to 2, this determines that the refinement will be applied to two global cells before decreasing by a level. Then select 'Blend to boundary depth -1', this forces the specified level of refinement to be used only in the subdivided cells that remain adjacent to the boundary. The remainder of the global cell will be refined at one level lower than the specified level. If 'Blend to Boundary Depth' is used then the entire global cell adjacent to the boundary will be sub-divided to the specified level. Further details can be found in the GEOMETRY chapter of the Documentation. The boundary refinement specification can be saved either in the triangle model file (.tri) or in the global mesh file (.mesh). Often it is more convenient to save the refinement in the mesh file as it can be more easily transferred to different models.



#	Show	# Tri	Type	Refine	
1	Yes	44924	Wall	No	bou
2	Yes	1640	Wall	No	bou
3	Yes	8	Inlet/Outle	No	bou
4	Yes	88	Inlet/Outle	No	bou

Paint Angle 45

Specification for boundary 2
Refinement depth at boundary 3
Refinement blending distance 2

Refinement Blending
- Blend to boundary depth
- [✓ Blend to boundary depth - 1]

Refinement Specification Destination
- Save specification in triangle file
- [✓ Save specification in mesh file]

```
14:21:27 - 23330 Nodes
14:21:27 - Model dimensions:-
  xmin = -0.153870   xmax = 0.193000
  ymin = 0.020499   ymax = 0.270500
  zmin = -0.036690   zmax = 0.326520
```

Figure 19.27: Setting Boundary Refinement

Save the mesh file as port.mesh and this then completes the mesh preparation.

19.2.9 Mesh Generation

In order to generate the computational mesh only the .tri and .mesh files are needed. The mesher, vmesh, can either be started from the command line or within R-desk. In this case R-desk will be used. Start the R-Desk GUI; either by typing rdesk at the command prompt or on Windows through the start menu. 'Start' > 'Programs' > 'Ricardo' > 'R-Desk'. By default R-Desk will open in a VIEWER project, which allows general visualisation of results for numerous Ricardo Software products, however in this case we require a VECTIS project, to give access to the mesh generation and solver setup menus. There close the current default VIEWER project ('File' > 'Close Project') and open a new project ('File' > 'Open' > 'Project'). The new project dialog box will appear. Here select VECTIS from the combo box and name your project 'port'.

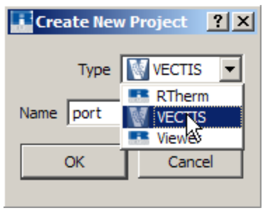
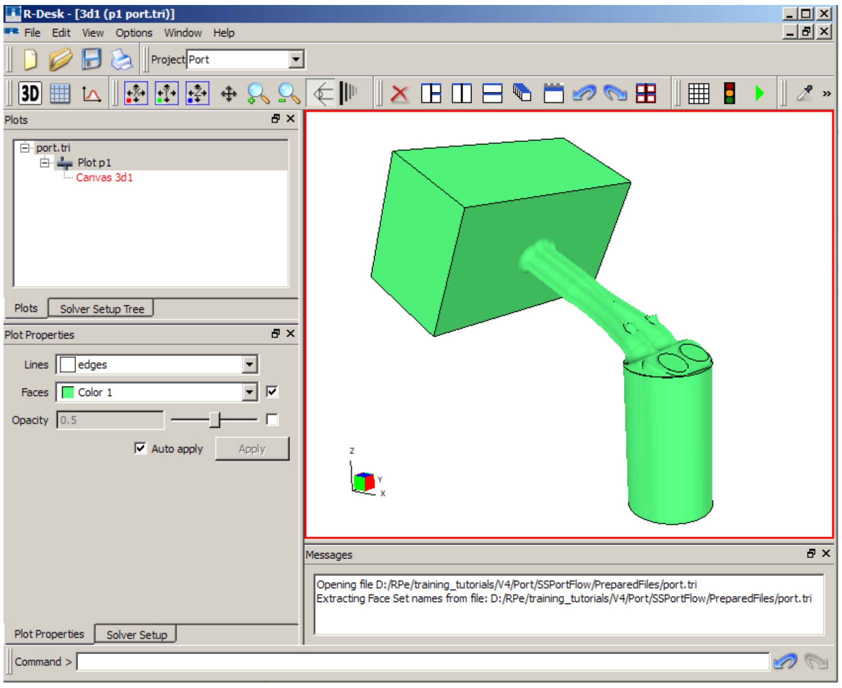


Figure 19.28: The New Project dialog box

Note. The default project type can be changed in the preference panel ('Options' > 'Preferences' > 'Miscellaneous'). Alternatively from the command line the project type can be specified using the '-project' switch (for example 'rdesk -project vectis').

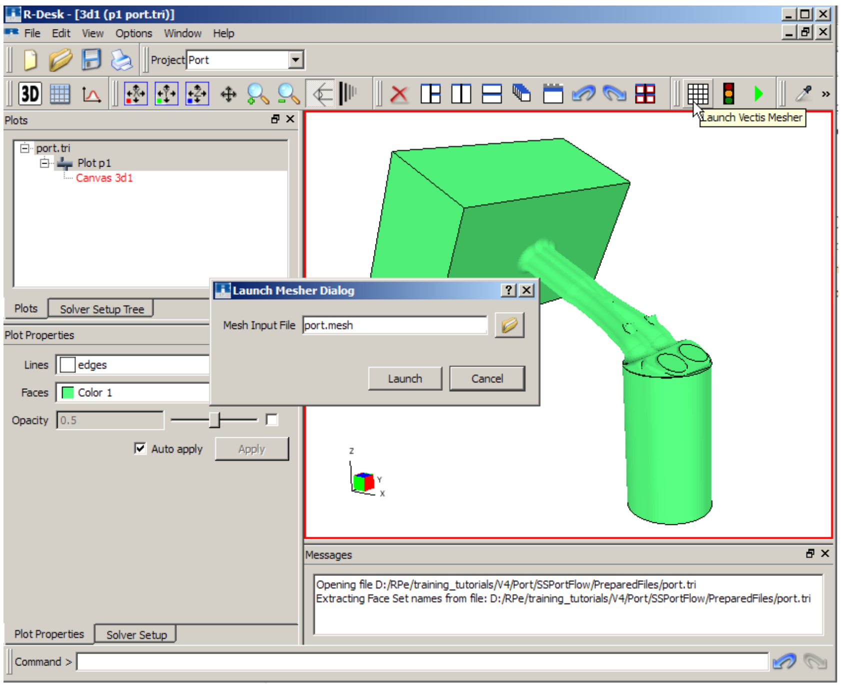
The geometry file can be opened and visualised. 'File' > 'Open' > 'File', then browse to the saved port.tri file. The geometry should open and then be plotted in a 3D canvas.



```
Opening file D:/RPe/training_tutorials/V4/Port/SSPortFlow/PreparedFiles/port.tri
Extracting Face Set names from file: D:/RPe/training_tutorials/V4/Port/SSPortFlow/PreparedFiles/port.tri
```

Figure 19.29: Visualising the geometry file in R-Desk

The mesher, vmesh, can be started from within the VECTIS project. Click on the launch mesher button Figure 19.30

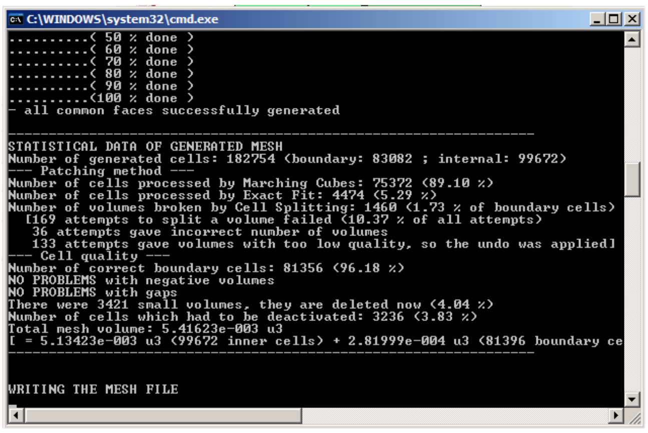


Mesh Input File: port.mesh

```
Opening file D:/RPe/training_tutorials/V4/Port/SSPortFlow/PreparedFiles/port.tri
Extracting Face Set names from file: D:/RPe/training_tutorials/V4/Port/SSPortFlow/PreparedFiles/port.tri
```

Figure 19.30: The Launch Mesher dialog box

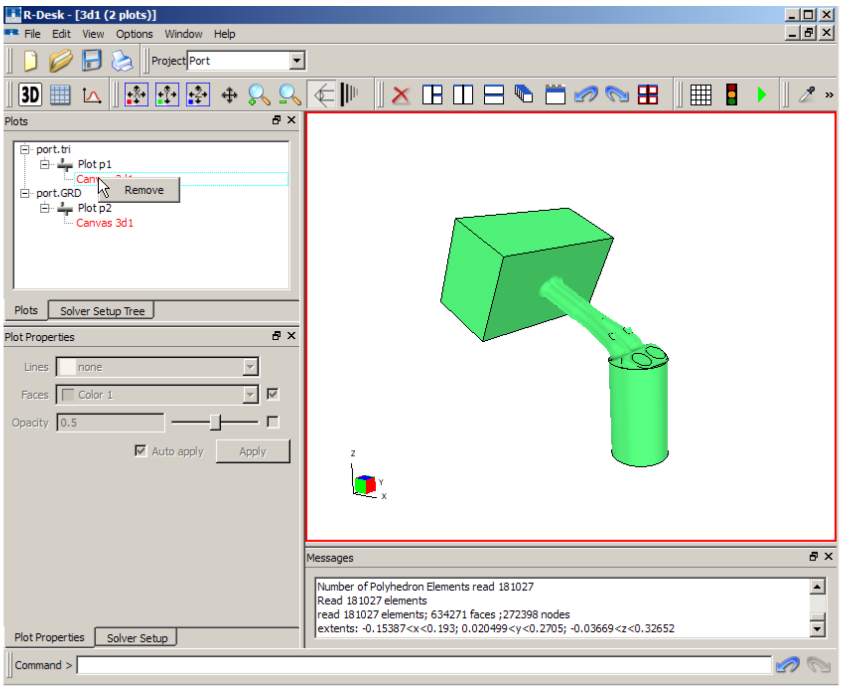
Browse to the .mesh file that was saved in phase1. Click on 'Launch' to start the mesher. A terminal window should open displaying the output from the mesher. Once the meshing is complete the terminal window will close. The screen output is also written to a file in the same directory as the .mesh file. The file should be called port.OUT. In this case the mesh will have around 200,000 cells.



```
..........( 50 % done )
..........( 60 % done )
..........( 70 % done )
..........( 80 % done )
..........( 90 % done )
..........(100 % done )
- all common faces successfully generated

-----------------------------------------------------------------
STATISTICAL DATA OF GENERATED MESH
Number of generated cells: 182754 (boundary: 83082 ; internal: 99672)
--- Patching method ---
Number of cells processed by Marching Cubes: 75372 (89.10 %)
Number of cells processed by Exact Fit: 4474 (5.29 %)
Number of volumes broken by Cell Splitting: 1460 (1.73 % of boundary cells)
  [169 attempts to split a volume failed (10.37 % of all attempts)
   36 attempts gave incorrect number of volumes
   133 attempts gave volumes with too low quality, so the undo was applied]
--- Cell quality ---
Number of correct boundary cells: 81356 (96.18 %)
NO PROBLEMS with negative volumes
NO PROBLEMS with gaps
There were 3421 small volumes, they are deleted now (4.04 %)
Number of cells which had to be deactivated: 3236 (3.83 %)
Total mesh volume: 5.41623e-003 u3
[ = 5.13423e-003 u3 (99672 inner cells) + 2.81999e-004 u3 (81396 boundary ce
-----------------------------------------------------------------


WRITING THE MESH FILE
```

Figure 19.31: Screen output from the mesher

The mesher will write the generated computational mesh to a grid file (In this case port.GRD). This can be opened in R-Desk for visualisation purposes. It is also possible to modify the boundary regions contained in the grid file within R-Desk, however this is outside the scope of this tutorial. Open the grid file in R-desk ('File' > 'Open' > 'File', then browse to port.GRD). This will be opened into the 3D canvas in the same way that the triangle file was. If both plots are found in the same canvas you may want to remove the port.tri plot from the canvas. Right click on the relevant canvas name found under the plot port.tri in the plot tree , and select remove . See Figure 19.32



```
Number of Polyhedron Elements read 181027
Read 181027 elements
read 181027 elements; 634271 faces ;272398 nodes
extents: -0.15387<x<0.193; 0.020499<y<0.2705; -0.03669<z<0.32652
```

Figure 19.32: Removing a plot from a canvas

To visualise the computational cells set the line attribute for the plot in the Plot Properties panel to Mesh. If the Plot Properties panel is not open, it can be opened by either through the following menu 'View' > 'Plot Properties' , right-clicking the header bar of a currently open panel (Figure 19.33) or by right-clicking the plot in the Plot Tree and selecting view > properties.

Make sure that the port.GRD model is selected in the Plot Tree then in the Plot Properties panel set the lines to 'mesh'. With mesh lines shown the computational grid should look something similar to that shown in Figure 19.34

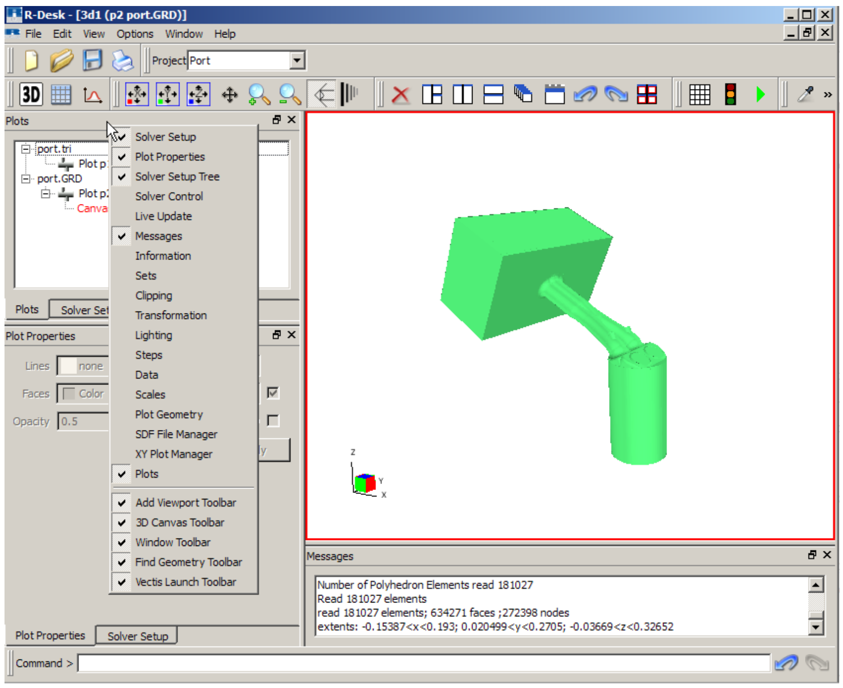
Now that the computational mesh has been generated we can move onto the next stage, setting up the input file for the solver.

19.2.10 Solver Setup

In order to run the solver the calculation inputs need to be defined. This includes the boundary conditions, fluid/solid properties, solver parameters, output data and frequency, initial conditions and so on...

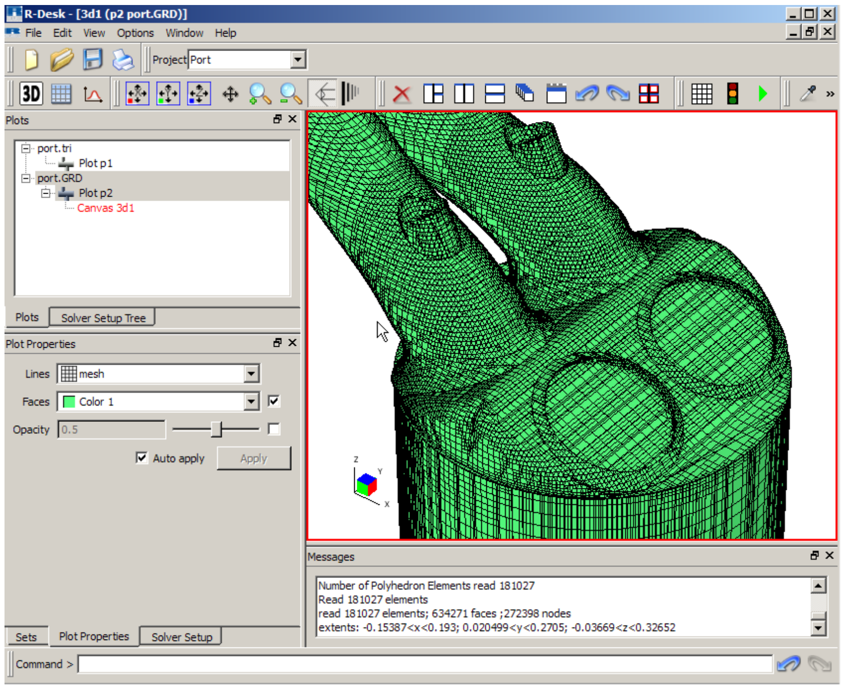
This is done using the Solver Setup tree and Solver Setup Input found within an R-Desk VECTIS project.

Firstly open up R-desk if it is not already open. Then using the project browser select 'New Project'. This will open the New project panel , select 'VECTIS' from the drop down list.



```
Number of Polyhedron Elements read 181027
Read 181027 elements
read 181027 elements; 634271 faces ;272398 nodes
extents: -0.15387<x<0.193; 0.020499<y<0.2705; -0.03669<z<0.32652
```

Figure 19.33: Drop down list allowing panels to be opened or closed



Lines mesh
Faces Color 1 ✓
Opacity 0.5
✓ Auto apply

```
Number of Polyhedron Elements read 181027
Read 181027 elements
read 181027 elements; 634271 faces ;272398 nodes
extents: -0.15387<x<0.193; 0.020499<y<0.2705; -0.03669<z<0.32652
```

Figure 19.34: Removing a plot from a canvas

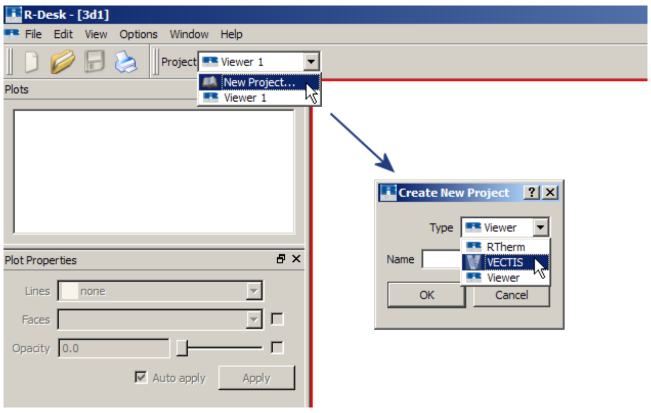


Figure 19.35: Open a new VECTIS project

The new project will open with the default layout as shown in Figure 19.36. The Solver Setup tree opens on the left hand side and Solver Setup Input should open on the right.

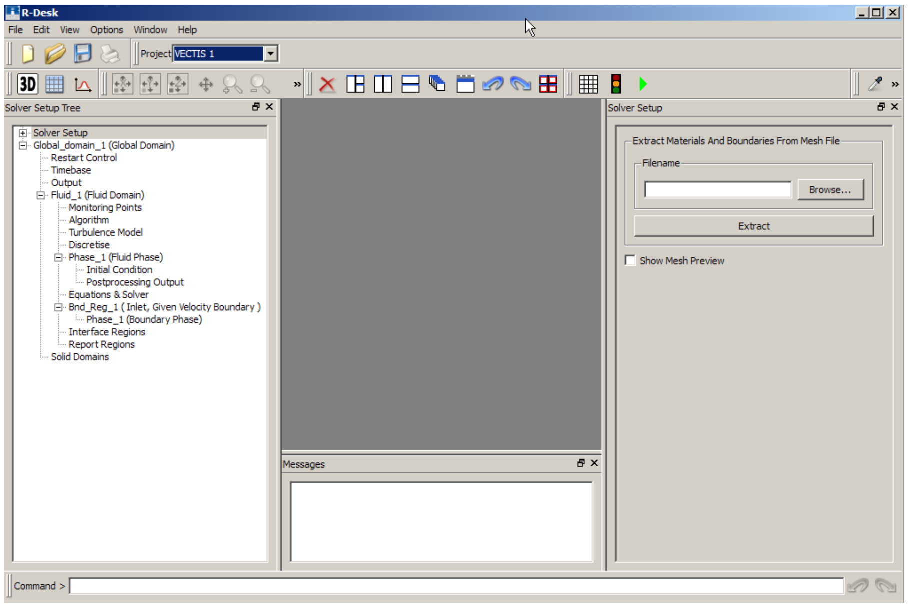


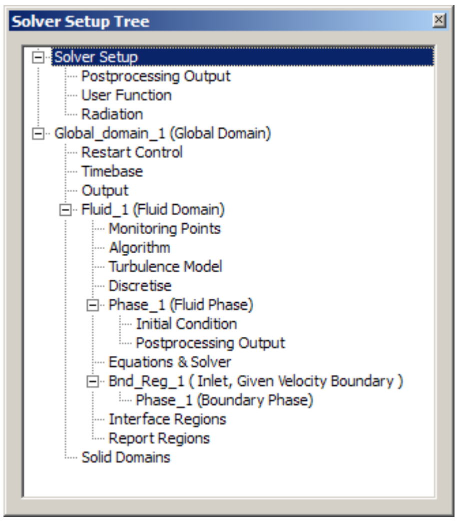
Figure 19.36: Default panel layout for VECTIS Project

The structure of the Solver Setup Tree reflects the general structure of calculation.

- Global domain containing general options for the calculation, timebase etc..
- Fluid domains containing data for phases, species and boundaries
- Solid domains containing materials and boundaries

The contents of the Solver Setup Input panel will change dynamically corresponding to the selected entry in the solver setup tree .

Click on the first entry in the tree 'Solver Setup'. The Solver Setup Input will show options to import the computational grid.



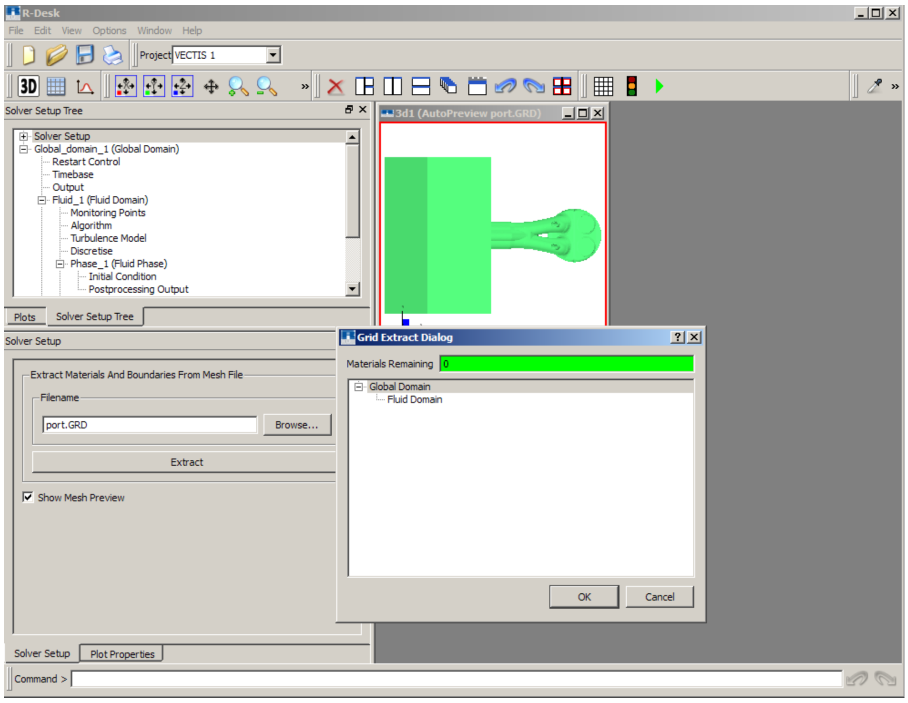
```
Solver Setup Tree
⊟ Solver Setup
    Postprocessing Output
    User Function
    Radiation
⊟ Global_domain_1 (Global Domain)
    Restart Control
    Timebase
    Output
  ⊟ Fluid_1 (Fluid Domain)
      Monitoring Points
      Algorithm
      Turbulence Model
      Discretise
    ⊟ Phase_1 (Fluid Phase)
        Initial Condition
        Postprocessing Output
      Equations & Solver
    ⊟ Bnd_Reg_1 ( Inlet, Given Velocity Boundary )
        Phase_1 (Boundary Phase)
      Interface Regions
      Report Regions
    Solid Domains
```

Figure 19.37: The solver setup input tree

19.2.10.1 Importing the Grid File

The materials and boundaries defined in a grid file can be imported into the solver setup. This will populate the solver setup tree with the correct number of materials and boundaries. A .GRD file needs to be entered into the 'Filename' box. Clicking on the browse button will open a dialog box and allow the relevant grid file to be selected. Once a file has been selected. The name will appear in the Filename box. Next, click the extract button.

A dialog box pops up and displays the materials found in the imported grid file.



```
⊞ Solver Setup
⊟ Global_domain_1 (Global Domain)
    Restart Control
    Timebase
    Output
  ⊟ Fluid_1 (Fluid Domain)
      Monitoring Points
      Algorithm
      Turbulence Model
      Discretise
    ⊟ Phase_1 (Fluid Phase)
        Initial Condition
        Postprocessing Output
```

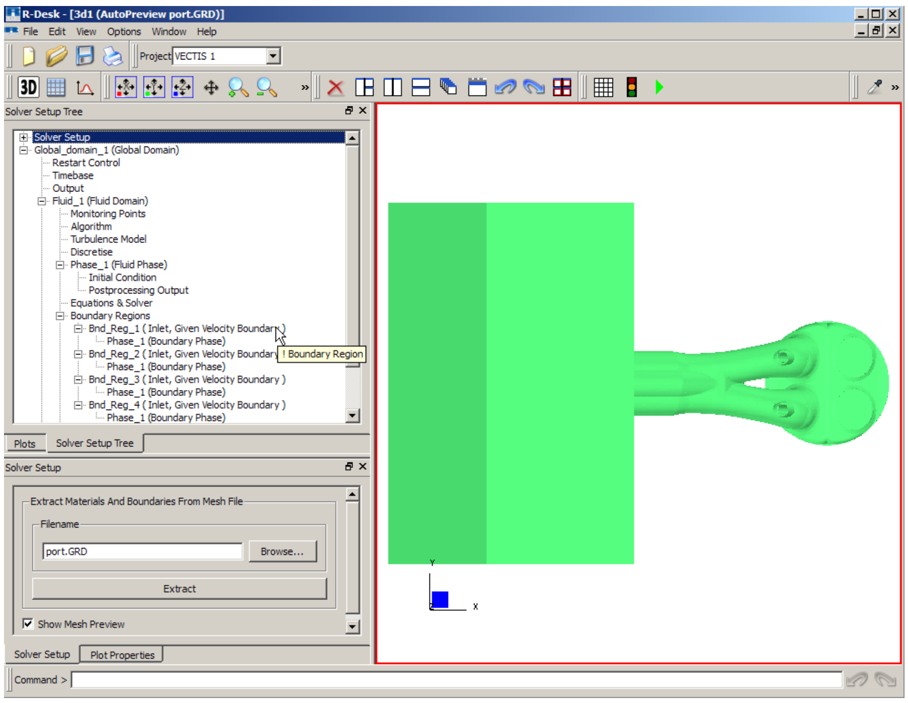
Extract Materials And Boundaries From Mesh File
Filename: port.GRD [Browse...]
[Extract]
☑ Show Mesh Preview

Materials Remaining: 0

```
⊟ Global Domain
    Fluid Domain
```

Figure 19.38: Importing the mesh into R-Desk Solver Setup

Each material in the grid file needs to be allocated as a fluid domain or as a material in a solid domain. In this case the computational grid contains a single fluid domain. As such the default setup for fluid/solid domains in the mesh import dialog is correct for this case.



Filename: port.GRD

☑ Show Mesh Preview

Figure 19.39: Grid file imported into R-Desk

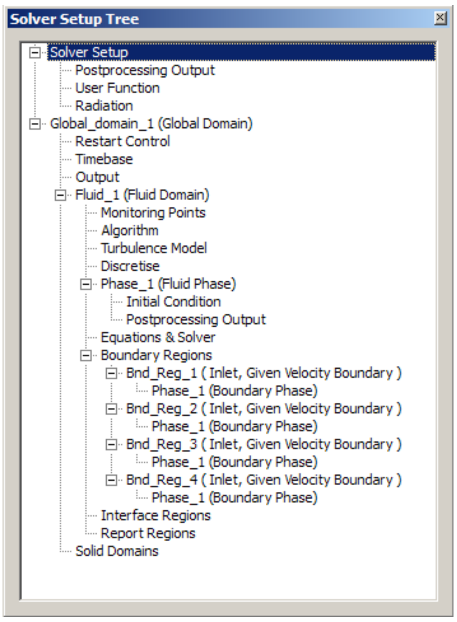
By importing the grid file the Solver Setup Tree is automatically populated with the correct number of domains and boundary regions. If show mesh preview is checked, the GRD file will be displayed in a 3D canvas. Then when entries in the Solver Setup Tree are selected the relevant domains, materials or boundaries regions of the grid are highlighted.checked, the GRD file will be displayed in a 3D canvas. Then when entries in the Solver Setup Tree are selected the relevant domains, materials or boundaries regions of the grid are highlighted.

19.2.10.2 Global Domain

Click on the global domain entry in the Solver Setup Tree . The global domain panel allows general simulation parameters to be set.

Domain Name A name can be given to each global domain in the calculation. For example ’Coolant’ or ’Cylinder Head’. In this case we will call the domain ’port’.

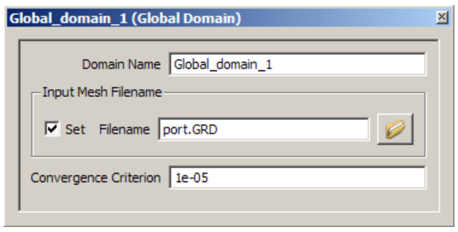
Input Mesh Filename The solver will default to using a computational grid file with the name projectname.GRD. A different grid file can be specified here. (otherwise the projectname is taken as the .inp file prefix). Again here we will use the .GRD file that was previously generated, normally port.GRD.



Solver Setup Tree

```
Solver Setup
    Postprocessing Output
    User Function
    Radiation
Global_domain_1 (Global Domain)
    Restart Control
    Timebase
    Output
    Fluid_1 (Fluid Domain)
        Monitoring Points
        Algorithm
        Turbulence Model
        Discretise
        Phase_1 (Fluid Phase)
            Initial Condition
            Postprocessing Output
        Equations & Solver
        Boundary Regions
            Bnd_Reg_1 ( Inlet, Given Velocity Boundary )
                Phase_1 (Boundary Phase)
            Bnd_Reg_2 ( Inlet, Given Velocity Boundary )
                Phase_1 (Boundary Phase)
            Bnd_Reg_3 ( Inlet, Given Velocity Boundary )
                Phase_1 (Boundary Phase)
            Bnd_Reg_4 ( Inlet, Given Velocity Boundary )
                Phase_1 (Boundary Phase)
        Interface Regions
        Report Regions
    Solid Domains
```

Figure 19.40: Solver setup input tree after grid import



Global_domain_1 (Global Domain)

Domain Name: Global_domain_1

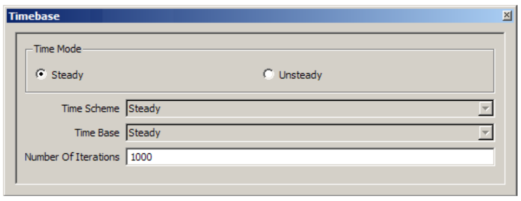
Input Mesh Filename: [✓ Set] Filename: port.GRD

Convergence Criterion: 1e-05

Figure 19.41: The Global Domain Panel

Timebase Click on the Timebase entry in the Solver Setup Tree . This allows the calculation timebase options to be defined. In this port case the 'Steady State' option will be used with a maximum of 1000 iterations

The other options available are detailed below.



Timebase

Time Mode: [✓ Steady] Unsteady

Time Scheme: Steady

Time Base: Steady

Number Of Iterations: 1000

Figure 19.42: The Timebase Panel

Time Mode Here the type of calculation time mode is selected

1. Steady
   - A maximum number of iterations is chosen for the calculation.
   - If all the residuals for all the equations are less than the specified tolerance the calculation will terminate.

2. Unsteady

Time Scheme For transient calculations this can be set to either '1st Order Euler' or '2nd Order Implicit'

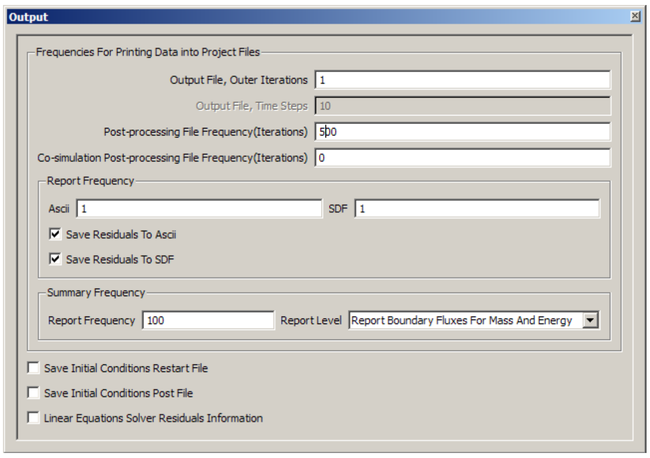
Time Base For transient calculations, the following timebases are available

1. 2-Stroke
2. Seconds
3. 4-Stroke
4. Non-Dimensional

Time Dependant Boundary Conditions The boundary conditions can be steady or unsteady/transient, which is determined by selecting the 'time dependent boundary conditions' toggle.

Timesteps The number of timesteps is chosen for the calculation, the end time being the chosen time step multiplied by the number of timesteps specified. Additionally, a 'Maximum Number of Outer Iterations' is chosen. This is the maximum number of iterations performed per time step.

- Typically the number of iterations required per step, will reduce after the initial period of the calculations. However this will be dependent on the accuracy of the applied initial conditions.



Output

Frequencies For Printing Data into Project Files

Output File, Outer Iterations	1
Output File, Time Steps	10
Post-processing File Frequency(Iterations)	500
Co-simulation Post-processing File Frequency(Iterations)	0

Report Frequency

Ascii: 1 SDF: 1

[✓] Save Residuals To Ascii

[✓] Save Residuals To SDF

Summary Frequency

Report Frequency: 100 Report Level: Report Boundary Fluxes For Mass And Energy

[ ] Save Initial Conditions Restart File

[ ] Save Initial Conditions Post File

[ ] Linear Equations Solver Residuals Information

Figure 19.43: The Solution Control Panel

Output

Output File, Outer Iterations/Time Steps Here the frequency of data reporting output is chosen. Firstly the monitoring and convergence data written to the screen is also written to the .out file at the intervals specified. If zero values are used then the output is not written.

Post-processing Frequency The frequency of post-processing file writing is specified in the 'Post-processing File Frequency' box.

Co-Simulation Post-processing Frequency Controls the frequency at which post-processing data is written when WAVE requests postprocessing output during a coupled simulation.

Additional output Additional output file can be written for monitoring data, boundary data, report regions and domain data. These can be either in ASCII column format or Ricardo SDF binary format. The frequency of the two types of file can be chosen independently by entering the required intervals in to the 'Report Frequency' boxes. Again an interval of zero will mean that the data is not written. The ASCII files include header rows detailing the data in each of the columns. Separate files are written for the different data types.

- Domain
- Monitoring
- IO and Wall (If reporting is selected for the relevant boundary)
- Arbitrary Surface

Additionally the residual data can be chosen to be saved to an ASCII file and the SDF report file.

The ASCII data files are used by the 'Live Update' utility in R-Desk. The SDF binary format is written to a single report file named projectname.rep_runnumber. It contains the data for all the additional output. It can be viewed and plotted in R-Desk using the 'SDF File Manager' and 'XY Plot Manager'

Summary output Summary data can also be written to the screen and output file at the chosen frequency and detail level.

Save Initial Conditions The 'Save Initial Conditions Restart File' and 'Save Initial Conditions Post File' toggles allow the user to select whether the initial data used in the calculations is saved to a file. The initial conditions are saved after any flow potential calculations are performed but before the first timestep/iteration.

Linear Equations Solver Residuals Information This toggles whether residual information is written for each of the equations at every timestep. This will write to the screen and output files in the format shown below.

Fluid Domain In this case the calculation will consist of one single-phase fluid domain

Domain Name A name can be assigned to the fluid domain to allow for easier reference later in the calculation.

Material Name A name can be assigned to each material in the fluid domain, again to allow for easier reference later in the calculation. Each fluid domain contains only one material

Material ID This refers to the material ID number referenced in the GRD file. Each separate material will have a unique ID number

Multiphase Modelling This allows the user to specify whether the fluid contains a single phase or a homogeneous mixture of a number of different phases.

Initial Values Initial velocity, pressure, temperature, passive scalar, turbulence and species mass fraction for the solution domain at the start time, as well as providing some other initial conditions options.

- Designed by User (This option allows initial conditions to be specified using a user routine)
- Potential Flow (This option will calculate the initial flow field based on the boundary condition data specified)
- Uniform

Potential Flow Scale Factor This allows the initial calculated potential field to be scaled. This is useful because the calculated potential flow may be much higher than the actual flow as turbulence and viscous effects are not taken into account.

Reference Data The reference values should have representative values for the solution.

Body Force Options to allow the simulation of body force.

The port flow uses air as the fluid so the default values should be suitable for the simulation

In this case the potential solver will be used to initialise the flow field.

The remaining fluid panels (algorithm, turbulence model, equations & solver and discretise) will be left at their default values.

Monitoring Points

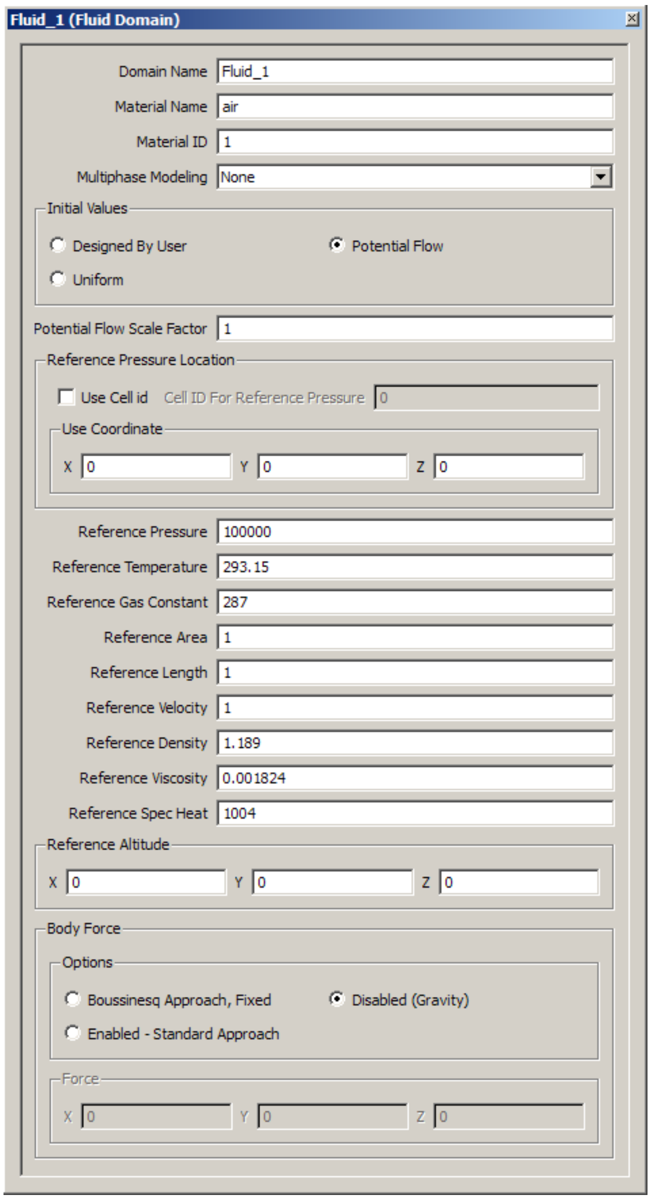
Monitoring Monitoring points are specified within the model domain where values of the solution are output at the end of each time step. A cell ID number or X,Y,Z co-ordinate can be specified for monitoring velocity values. Additional monitoring points can be specified using the Add button. Values for the first monitoring point are written to the screen. Values for all of the monitoring points are written to files if reporting options are set in the Output Panel

Next the fluid phase needs to be set-up.

Fluid Phase The fluid phase panel contains data for each fluid phase. Each phase may be made up of a single or multiple species.

Phase Name Each phase can be given a name to aid reference.

Mixture of Species Option Each phase can be made up of a number of different components/species. In this tutorial a single-component phase is considered. More details of the different options can be found in 'Setting a multiphase mixture'.



Fluid_1 (Fluid Domain)

Domain Name	Fluid_1
Material Name	air
Material ID	1
Multiphase Modeling	None

Initial Values

Designed By User [✓ Potential Flow]

Uniform

Potential Flow Scale Factor	1

Reference Pressure Location

Use Cell id Cell ID For Reference Pressure 0

Use Coordinate

X 0 Y 0 Z 0

Reference Pressure	100000
Reference Temperature	293.15
Reference Gas Constant	287
Reference Area	1
Reference Length	1
Reference Velocity	1
Reference Density	1.189
Reference Viscosity	0.001824
Reference Spec Heat	1004

Reference Altitude

X 0 Y 0 Z 0

Body Force

Options

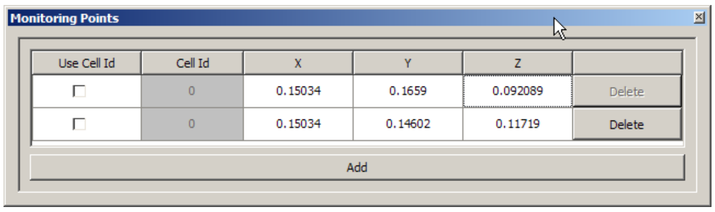
Boussinesq Approach, Fixed [✓ Disabled (Gravity)]

Enabled - Standard Approach

Force

X 0 Y 0 Z 0

Figure 19.44: The Fluid Domain Panel

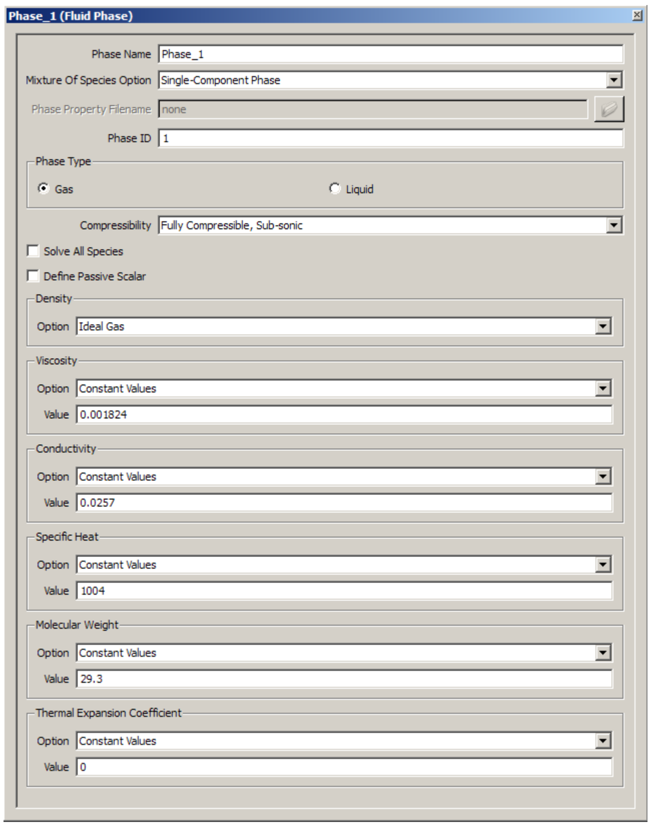


Monitoring Points

Use Cell Id	Cell Id	X	Y	Z	
☐	0	0.15034	0.1659	0.092089	Delete
☐	0	0.15034	0.14602	0.11719	Delete

Add

Figure 19.45: The Monitoring pint panel


Phase_1 (Fluid Phase)

Phase Name: Phase_1
Mixture Of Species Option: Single-Component Phase
Phase Property Filename: none
Phase ID: 1

Phase Type
[✓ Gas] Liquid

Compressibility: Fully Compressible, Sub-sonic
☐ Solve All Species
☐ Define Passive Scalar

Density
Option: Ideal Gas

Viscosity
Option: Constant Values
Value: 0.001824

Conductivity
Option: Constant Values
Value: 0.0257

Specific Heat
Option: Constant Values
Value: 1004

Molecular Weight
Option: Constant Values
Value: 29.3

Thermal Expansion Coefficient
Option: Constant Values
Value: 0

Figure 19.46: Fluid Phase Options

Phase Type and compressibility The phase type can be selected to be either a gas or a liquid. In the case of liquid simulation the compressibility is set to be incompressible. For gases there is a choice of compressibility options;

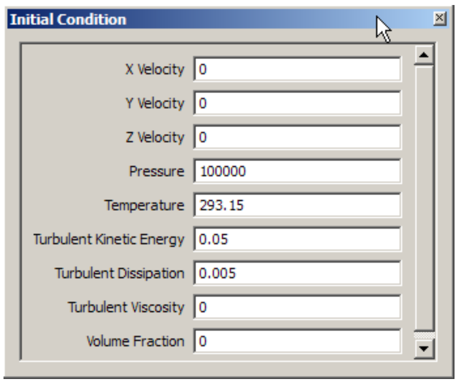
- Incompressible Fluid,
- Weakly Compressible
- Fully Compressible, Subsonic
- Fully Compressible, Supersonic

For the tutorial we specify a fully compressible, subsonic gas. More details of the different options can be found in 'Setting a phase and its properties'.

Solve all species This determines whether the mass fraction equations for all species are solved or whether the last species is taken as the remaining mass fraction after the other species are solved.

Property Specification In general there are a number of ways of specifying the properties; constant values or relationships.

Initial Conditions The initial conditions for the fluid can be specified in the initial conditions panel. This allows the user to specify initial flow velocity, pressure, temperature etc...



X Velocity	0
Y Velocity	0
Z Velocity	0
Pressure	100000
Temperature	293.15
Turbulent Kinetic Energy	0.05
Turbulent Dissipation	0.005
Turbulent Viscosity	0
Volume Fraction	0

Figure 19.47: Fluid Phase Initial Conditions

These values will be imposed on the fluid when uniform initial conditions are selected in the Fluid Domain panel

PostProcessing Output The data to be output to the post file can be selected in the output panel. Each of the scalar variables checked will be written to the file.

Boundary Regions Each of the boundary regions can then be specified.

Region Name A name can be given to each boundary for reference. The name can be referenced when using user programming routines.

Boundary report This specifies whether output data is written to the report files for this boundary

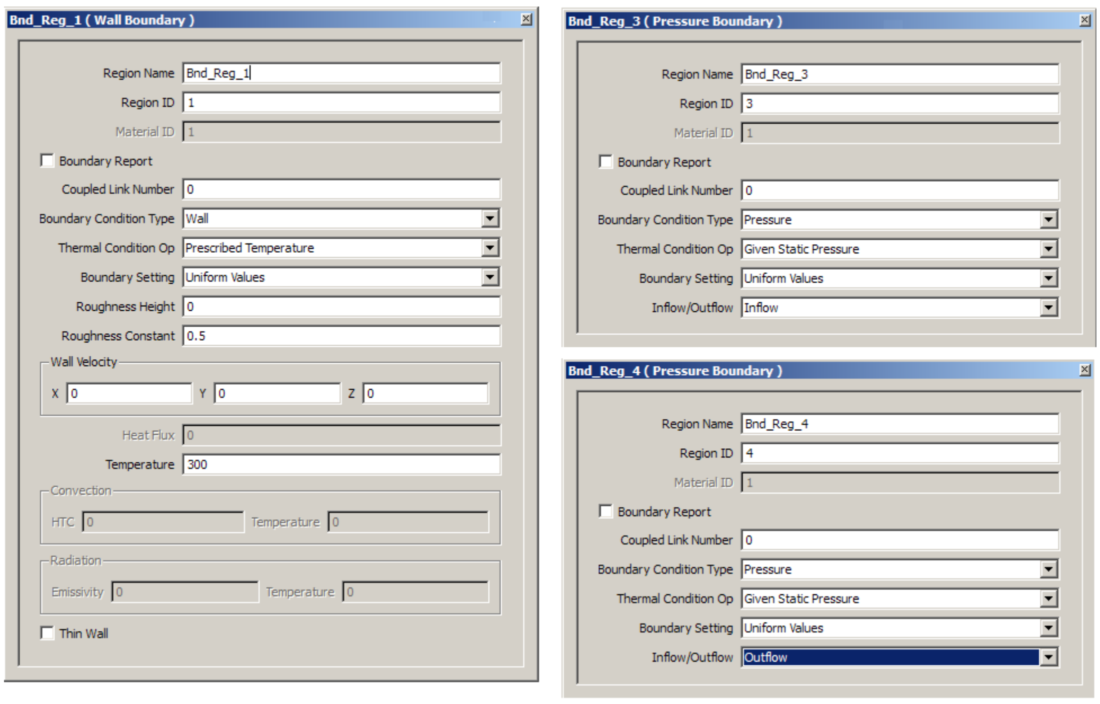
Coupled Link Number When running coupled WAVE/VECTIS the interface numbers for each coupled boundary are specified here.

Boundary Condition Type Next the appropriate boundary type is chosen for each boundary

Details of the different boundary conditions can be found in the BOUNDARY CONDITION TYPES chapter.

The following boundary conditions will be used.

- Boundaries 1 and 2 will be walls with a fixed temperature of 300 degK
- Boundary 3 will be a static pressure boundary set to 100000 Pa, inflow and with a temperature of 300 deg K
- Boundary 4 will be a static pressure boundary set to 93133 Pa, outflow and with a temperature of 300 deg K



Bnd_Reg_1 (Wall Boundary)

Region Name	Bnd_Reg_1
Region ID	1
Material ID	1
☐ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Wall
Thermal Condition Op	Prescribed Temperature
Boundary Setting	Uniform Values
Roughness Height	0
Roughness Constant	0.5

Wall Velocity: X 0 Y 0 Z 0

Heat Flux	0
Temperature	300

Convection: HTC 0 Temperature 0

Radiation: Emissivity 0 Temperature 0

☐ Thin Wall

Bnd_Reg_3 (Pressure Boundary)

Region Name	Bnd_Reg_3
Region ID	3
Material ID	1
☐ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Pressure
Thermal Condition Op	Given Static Pressure
Boundary Setting	Uniform Values
Inflow/Outflow	Inflow

Bnd_Reg_4 (Pressure Boundary)

Region Name	Bnd_Reg_4
Region ID	4
Material ID	1
☐ Boundary Report	
Coupled Link Number	0
Boundary Condition Type	Pressure
Thermal Condition Op	Given Static Pressure
Boundary Setting	Uniform Values
Inflow/Outflow	Outflow

Figure 19.48: Setting up the boundary regions

Report Regions Report Regions can be specified to allow data to be extracted on arbitrarily defined surfaces. The surfaces are defined by triangulated surfaces in the Ricardo standard data file (SDF) format typically called .tri or .sdf.

The arbitrary surface reports integrated quantities for the surface defined by the triangle file into the .rep sdf file and optionally into text output files. Values reported are similar to those for IO files.

In this case an arbitrary surface will be defined within the port to allow the angular momentum to be monitored during the simulation. A straightforward way to define the surface is to chop the geometry in phase1 at the location where the arbitrary surface is required.

Open the port.tri file in phase1. Rotate the model to view the outlet cylinder. Using the horizontal slice button, slice the model in the desired position, see Figure 19.49.

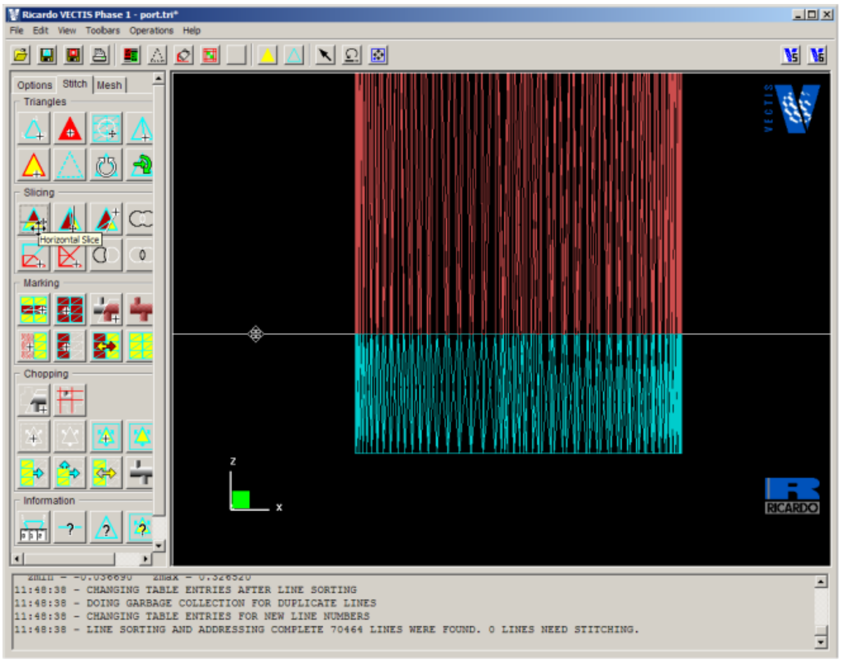
The model will be sliced with one half of the model marked in red, Figure 19.50.

Use the delete marked area button to delete the marked triangles. This will leave just the unmarked area of the model, Figure 19.50.

Use the cap hole tool to fill the hole, this will generate a surface in the location where the geometry was sliced, Figure 19.52.

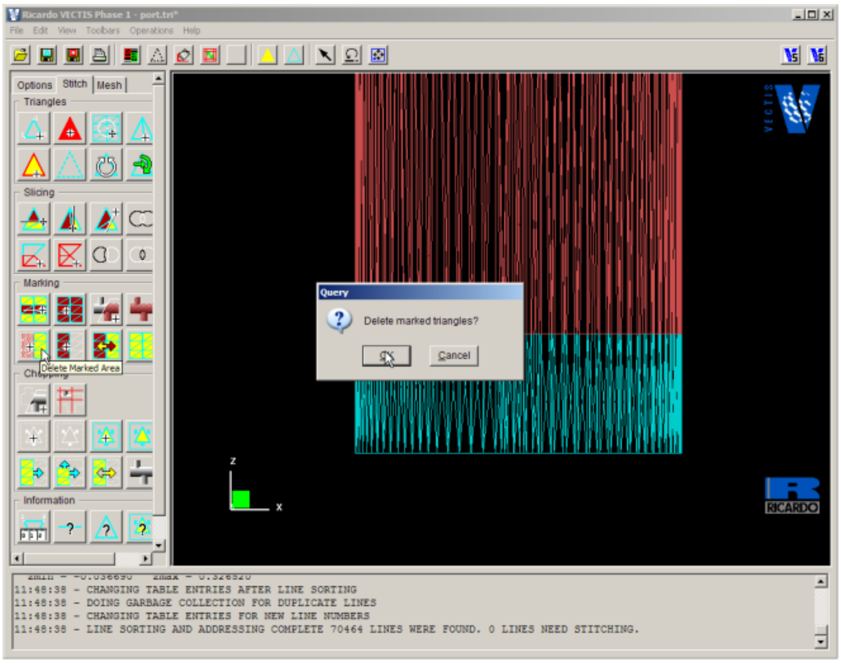
Then with the chop geometry tool select the triangles that make up the newly created surface, Figure 19.53.

Finally save the model as a new file ('File' > 'Save As'). When prompted select save only the active triangles. This will save only the triangles currently displayed, which should all belong to the arbitrary surface. Figure 19.54.



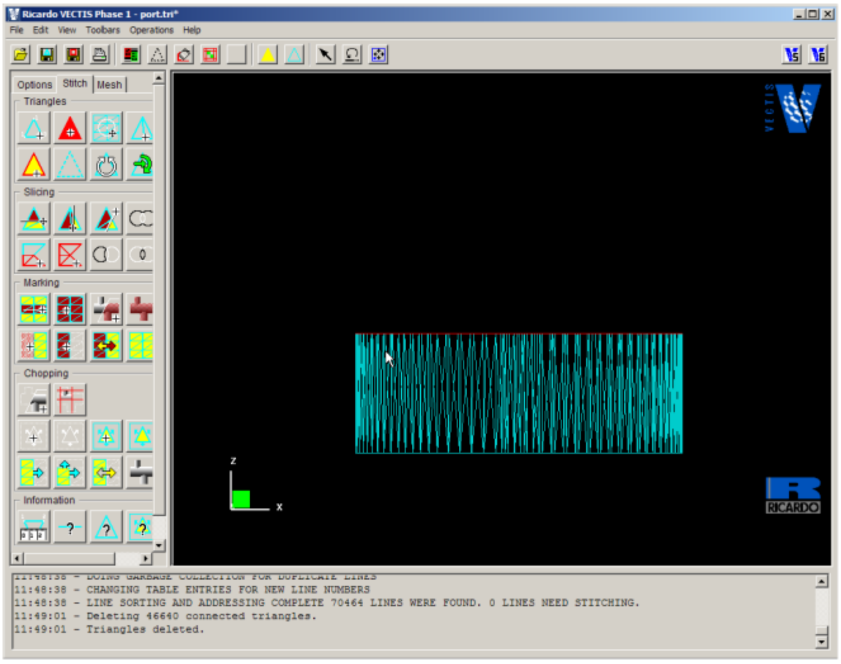
```
  zmin = -0.036690   zmax = 0.326520
11:48:38 - CHANGING TABLE ENTRIES AFTER LINE SORTING
11:48:38 - DOING GARBAGE COLLECTION FOR DUPLICATE LINES
11:48:38 - CHANGING TABLE ENTRIES FOR NEW LINE NUMBERS
11:48:38 - LINE SORTING AND ADDRESSING COMPLETE 70464 LINES WERE FOUND. 0 LINES NEED STITCHING.
```

Figure 19.49: Cut the model in the location of the arbitrary surface is required



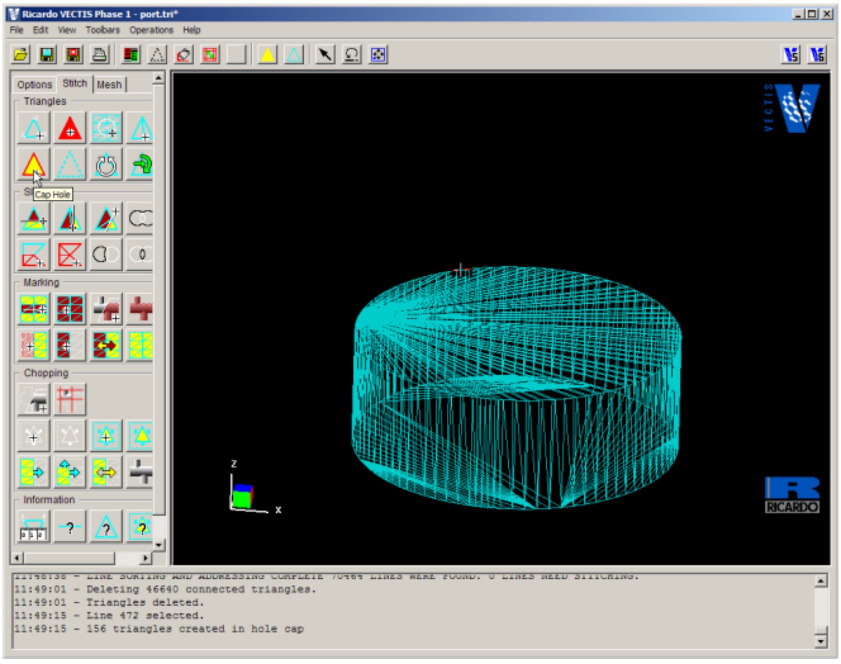
```
  zmin = -0.036690   zmax = 0.326520
11:48:38 - CHANGING TABLE ENTRIES AFTER LINE SORTING
11:48:38 - DOING GARBAGE COLLECTION FOR DUPLICATE LINES
11:48:38 - CHANGING TABLE ENTRIES FOR NEW LINE NUMBERS
11:48:38 - LINE SORTING AND ADDRESSING COMPLETE 70464 LINES WERE FOUND. 0 LINES NEED STITCHING.
```

Figure 19.50: Delete the marked area


Ricardo VECTIS Phase 1 - port.tri*
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Slicing
Marking
Chopping
Information
z
x
RICARDO

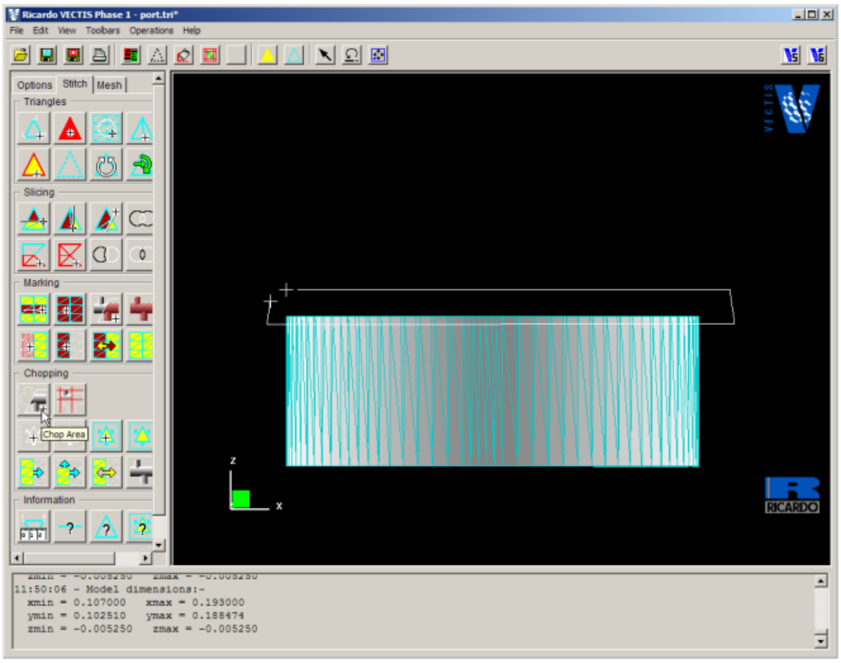
```
11:48:38 - DOING GARBAGE COLLECTION FOR DUPLICATE LINES
11:48:38 - CHANGING TABLE ENTRIES FOR NEW LINE NUMBERS
11:48:38 - LINE SORTING AND ADDRESSING COMPLETE 70464 LINES WERE FOUND. 0 LINES NEED STITCHING.
11:49:01 - Deleting 46640 connected triangles.
11:49:01 - Triangles deleted.
```

Figure 19.51: Model cut at location of arbitrary surface


Ricardo VECTIS Phase 1 - port.tri*
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Cap Hole
Marking
Chopping
Information
z
x
RICARDO

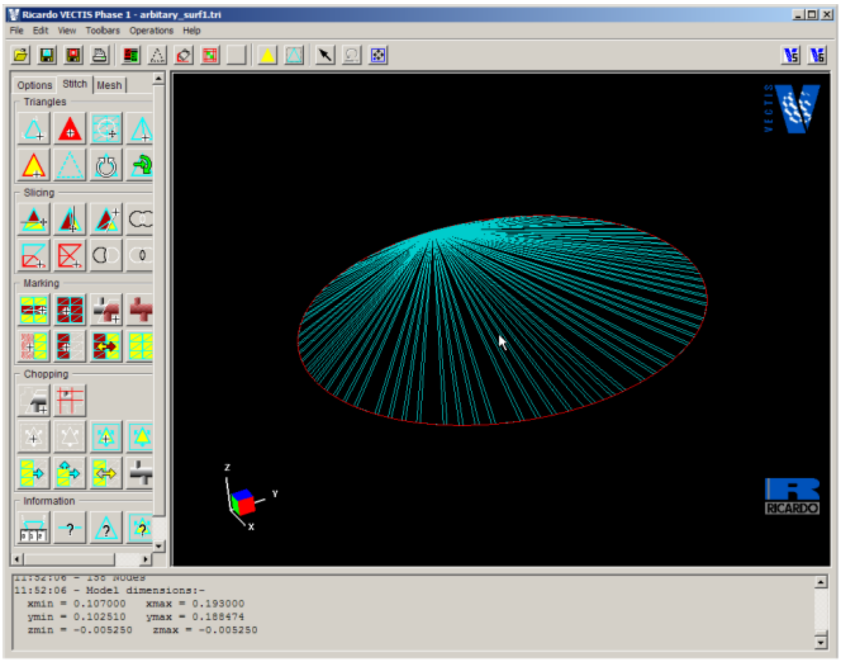
```
11:48:38 - LINE SORTING AND ADDRESSING COMPLETE 70464 LINES WERE FOUND. 0 LINES NEED STITCHING.
11:49:01 - Deleting 46640 connected triangles.
11:49:01 - Triangles deleted.
11:49:15 - Line 472 selected.
11:49:15 - 156 triangles created in hole cap
```

Figure 19.52: Using the cap hole tool to fill the open edge



```
  zmin = -0.005250   zmax = -0.005250
11:50:06 - Model dimensions:-
  xmin = 0.107000   xmax = 0.193000
  ymin = 0.102510   ymax = 0.188474
  zmin = -0.005250   zmax = -0.005250
```

Figure 19.53: Chop out the required surface


Ricardo VECTIS Phase 1 - arbitary_surf1.tri
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Slicing
Marking
Chopping
Information
z
y
x
RICARDO

```
11:52:06 - 138 Nodes
11:52:06 - Model dimensions:-
  xmin = 0.107000   xmax = 0.193000
  ymin = 0.102510   ymax = 0.188474
  zmin = -0.005250   zmax = -0.005250
```

Figure 19.54: Final Arbitrary Surface

The arbitrary surface does not have to be planar, it can be any form.

Once the surface has been defined then the report region in the solver setup tree can be setup. Select 'Report Regions' in the solver Setup Tree . Then add a report region using the add child report regions button.

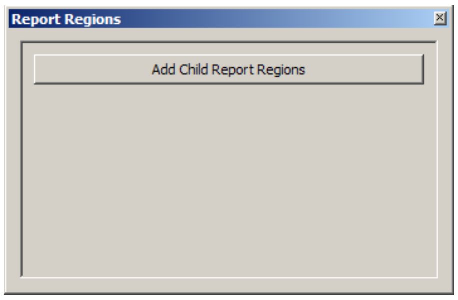
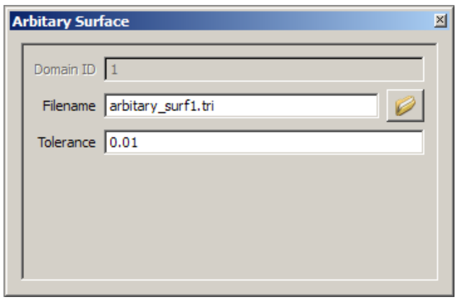


Figure 19.55: Add report region

The geometry filename needs to be specified along with a tolerance.


Arbitary Surface

Domain ID 1

Filename arbitary_surf1.tri

Tolerance 0.01

Figure 19.56: Specify triangle filename

The defined triangles are mapped to the solver mesh. Each triangle is successively divided along its longest edge until one of the following criteria are reached:

- All nodes of a triangle are contained within the same cell.
- All nodes of triangle are greater than, or less that the mesh extents (i.e. triangle is outside mesh)
- Longest edge is less than the user specified edge tolerance.

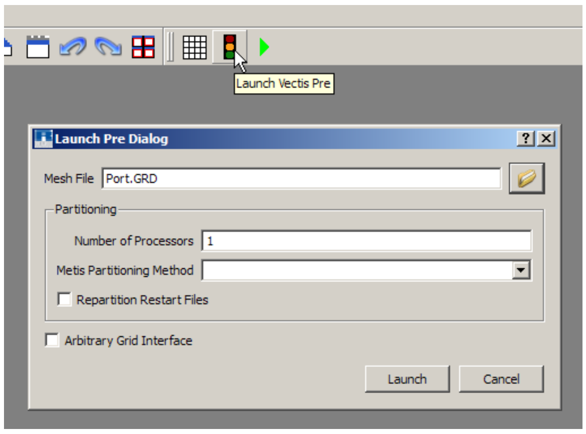
The input file should now be complete for the analysis Save the file as port.inp ('file' > 'save as')

19.2.11 Grid Preparation

Vpre prepares the computational grid for use in the solver. It can be started either from a cmd window or from *R-desk*. In a command window type 'vpre port.GRD' to run the *vpre* mesh pre-processor.

From *R-Desk* the Launch Vectis Pre button can be used to start *vpre*. Click on the button to open the 'Launch vpre' dialog box. Insert the name of the computational grid file in the 'mesh file' box.

Figure 19.57.



Launch Pre Dialog

Mesh File Port.GRD

Partitioning

Number of Processors 1

Metis Partitioning Method

☐ Repartition Restart Files

☐ Arbitrary Grid Interface

Launch Cancel

Figure 19.57: Launch Vpre

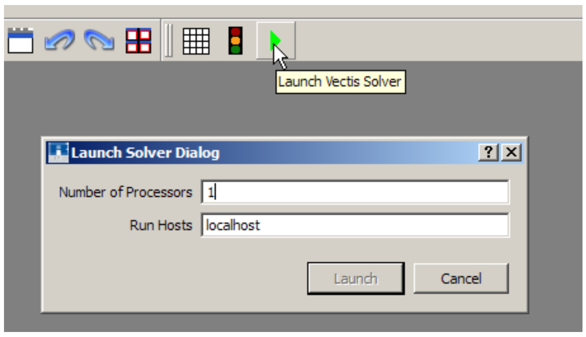
vpre can also be used to re-partition the mesh for parallel calculations.

19.2.12 Running the solver

Now that the .inp file has been created and the grid file has been processed the solver can be run.

Again this can either be run from the cmd window or from R-Desk. From the cmd window, type 'vsolve port.inp'.

From *R-Desk* the Launch Vectis Solver button can be used to start *vsolve*. Click on the button to open the 'Launch vpre' dialog box. Insert the run directory and the name of the inp file. Figure 19.58.



Launch Solver Dialog

Number of Processors 1

Run Hosts localhost

Launch Cancel

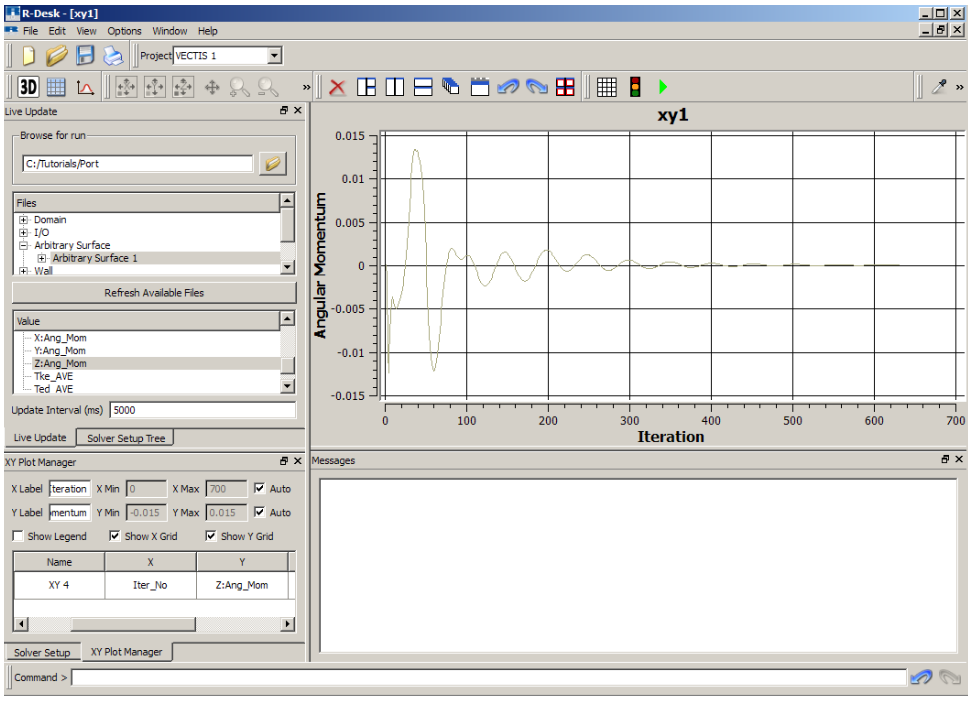
Figure 19.58: Launch Vsolve

The solver will start

19.2.13 Live Update

Live update is a utility in R-Desk that allows a simulation to be monitored whilst it is running. Firstly open a xy canvas using the new XY canvas button. Then in the Live Update panel, browse to the directory where the simulation is running. The available data files are presented in the 'Files' window. The data files available are determined by the ascii files selected in the reporting section of the 19.2.10.2Global Domain in the solver setup tree. Once a file is selected the data available to be plotted is shown in the 'Value' window. The different data can be dragged onto a xy plot canvas The xy plot can then be further modified using the XY plot manager

In this case the in-plane swirl can be plotted for the arbitrary surface. Click on the arbitrary surface in the files panel, then drag the 'Iter_no' entry onto an XY canvas followed by the 'Z:Ang_Mom' entry. This will show how the swirling motion converges throughout the simulation. In this case as the port is symmetric then the swirl around the cylinder axis should be close to zero. Figure 19.59.



Project VECTIS 1

C:/Tutorials/Port

Files
- Domain
- I/O
- Arbitrary Surface
  - Arbitrary Surface 1
- Wall

Value
- X:Ang_Mom
- Y:Ang_Mom
- Z:Ang_Mom
- Tke_AVE
- Ted_AVE

Update Interval (ms) 5000

Live Update | Solver Setup Tree

XY Plot Manager

X Label Iteration X Min 0 X Max 700 ✓ Auto
Y Label mentum Y Min -0.015 Y Max 0.015 ✓ Auto
☐ Show Legend ✓ Show X Grid ✓ Show Y Grid

Name	X	Y
XY 4	Iter_No	Z:Ang_Mom

Solver Setup | XY Plot Manager

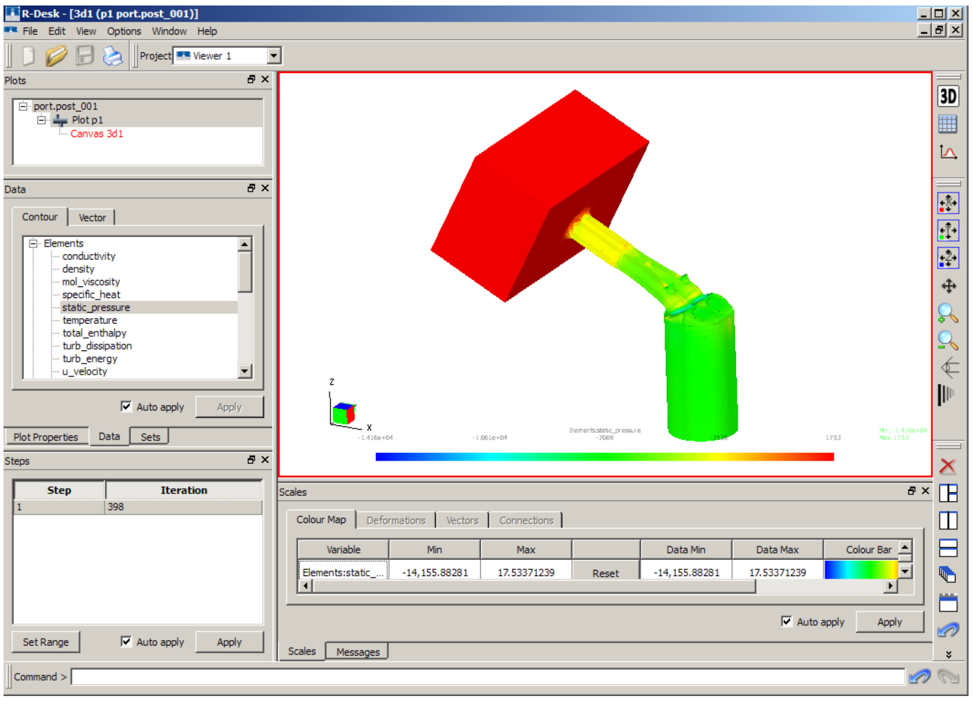
Figure 19.59: Live Update

19.2.14 Post-processing

Once the simulation converges it will stop and write restart and post-processing files. The projectname.post_0?? file contains simulation data for the entire calculation domain at specified post-processing times. Open the post file in R-Desk using the *'File'* > *'Open'* > *'File'* menu option. It should appear in the plot tree. By default the plot will be added to the active 3D canvas. Note: this behaviour can be changed in the preferences panel.

The attributes of the plot can be modified in the Plot Properties panel

Different data can be shown on the plot by selecting it in the Data panel



Step	Iteration
1	398

Variable	Min	Max		Data Min	Data Max	Colour Bar
Elements:static_...	-14,155.88281	17.53371239	Reset	-14,155.88281	17.53371239	

Figure 19.60: Pressure variation at the model surface

19.2.14.1 2D plane

A 2D slice can be made of a plot. Right-click the plot in the plot tree and select slice. The slice is then defined in the slice definition panel

If the plot is present in a 3D canvas the slice will be previewed. Once the slice is in the correct position, click 'apply' if 'auto apply' is not active. Then open an additional 3D canvas. The canvases can be tiled using the 'window' > 'Tile... ' options

Drag the slice plot into the second canvas. The data plotted on the slice can be changed using the 'data' panel

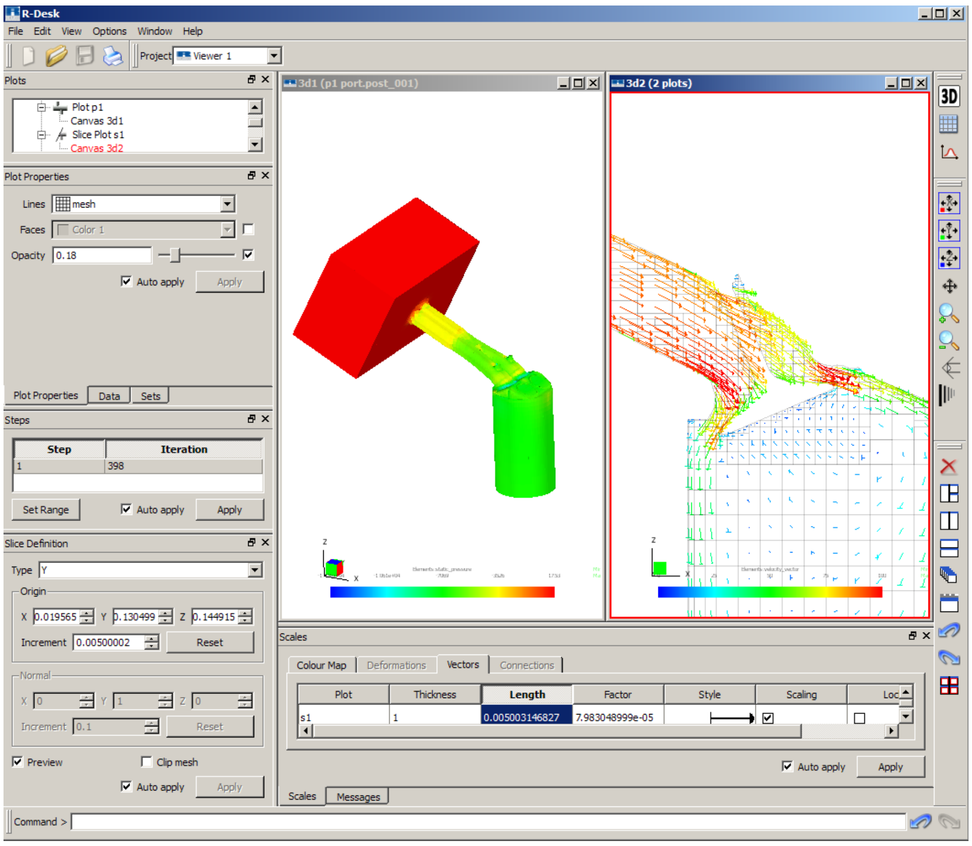
More information on the general post-processing capabilities can be found in the R-Desk help.

19.2.15 Initialisation Tips

In general, correct initialisation of the calculation will give faster convergence. In some cases were the initialisation of the calculation is poor, the simulation will diverge very quickly after starting.

Frequently, there are a few things that can be done to improve the initialisation of port flow cases, particularly when the flow rate is high.

When using the potential flow solver with pressure boundary conditions on both the inlet and outlet, the potential scale factor may need to be reduced . The potential solver neglects any viscous effects and the initial mass flow can therefore be over predicted. A value of 0.7 to 0.8 is usually sufficient to give a more stable initialisation although lower values may be necessary in extreme cases.



Plot Properties: Lines: mesh; Faces: Color 1; Opacity: 0.18

Steps:

Step	Iteration
1	398

Slice Definition: Type: Y

Origin: X 0.019565, Y 0.130499, Z 0.144915; Increment 0.00500002

Normal: X 0, Y 1, Z 0; Increment 0.1

Vectors:

Plot	Thickness	Length	Factor	Style	Scaling	Loc
s1	1	0.005003146827	7.983048999e-05	⟶	✓	☐

Figure 19.61: Slice plot showing velocity vectors

When using the potential flow solver with a mass flow and pressure boundary condition the pressure value on the mass flow boundary may need to be modified to reduce the specified pressure drop between the two boundaries.

When using potential flow initialisation it is important to ensure that the reference values specified are representative as they are used by the potential solver. Additional the temperature specified in the initial conditions section is also used by the potential flow solver and therefore must also be a representative value.

19.3 Coolant Flow

Get the necessary files for the coolant flow tutorial

http://www.software.ricardo.com/support/tutorials/vectismax/TutorialFiles
CoolantFlow

19.3.1 Introduction

The purpose of this tutorial is to illustrate the steps needed to set-up a simple engine coolant flow analysis. However the general process could also be applied to any incompressible flow simulation.

The automatic mesh generation capability of VECTIS-MAX makes the CFD analyses of engine coolant jackets particularly quick since the complex computational coolant jacket mesh is created very quickly from the closed CAD STL geometry. This allows investigations such as gasket hole position and size optimisation to be integrated into Engineering Development projects. It is also a common technique to perform coolant analysis and then extract the surface Heat Transfer Coefficient results and apply these to a Finite Element analysis as thermal boundary conditions.

CFD analyses of coolant jackets require boundary conditions for the inlets and outlets and the wall surfaces. The inlet and outlet boundary conditions are not usually an issue since the coolant mass flow rate and/or inlet and outlet pressure is usually known. However the coolant jacket surface temperature distributions are not usually known. There are several options to overcome this issue:

1. Run the coolant jacket analysis as a fluid only analysis and as an iso-thermal analysis such that both the coolant fluid temperature and the wall temperatures are the same. The Heat Transfer Coefficients (HTCs) are then mapped on to a Finite Element (FE) analysis mesh and used as thermal boundary conditions for the FE analysis. The predicted wall surface temperatures from the FE analysis can then be extracted from the FE analysis results and mapped on to the VECTIS computational mesh. The VECTIS CFD analysis is then run again but this time has realistic wall temperatures. For general gasket development and coolant jacket development it is common to simply perform the iso-thermal analysis and use these results without considering the wall temperature effects.

2. Run the coolant jacket analysis as a fluid only analysis and as an iso-thermal analysis until a steady state converged solution is achieved. Then use this fluid only analysis result to start a VECTIS conjugate analysis where both the fluid and the solid domains are solved in VECTIS as well as the heat flux between them and the surface wall temperatures. This is more accurate than option 1 and generally easier as all the thermal analysis work is performed in VECTIS. Both the surface and internal thermal results can then be mapped onto an FE mesh if required.

3. The calculation can be extended to include the solid components around the coolant water jacket using a conjugate heat transfer analysis. In this case the fluid will be solved non-isothermally. This will calculate the temperature distribution in the solid components and fluid taking into account the variation in heat transfer and wall temperature between the fluid and solid.

19.3.2 Aims

This tutorial assumes that the user has the required knowledge to use VECTIS-MAX to a level that is achieved after completing the basic tutorial. After completing this tutorial the user should have knowledge of the following:

- □ Setting up the coolant jacket geometry.
- □ Setting up the control mesh for coolant jacket analyses.
- □ Setting up the VECTIS-MAX solver for a coolant Jacket analysis.
- □ In particular how to set up the calculation to model incompressible liquid flows.
- □ Monitoring the analysis and convergence using Live Update.
- □ Postprocessing the results.

19.3.3 Geometry Preparation

Phase 1 is the current pre-processing package for use with VECTIS-MAX. Phase 1 is used to read in geometry and process it to a form acceptable to the VECTIS-MAX mesh generator. Depending on the quality of the initial geometry a certain amount of repair may be needed to form a single closed volume.

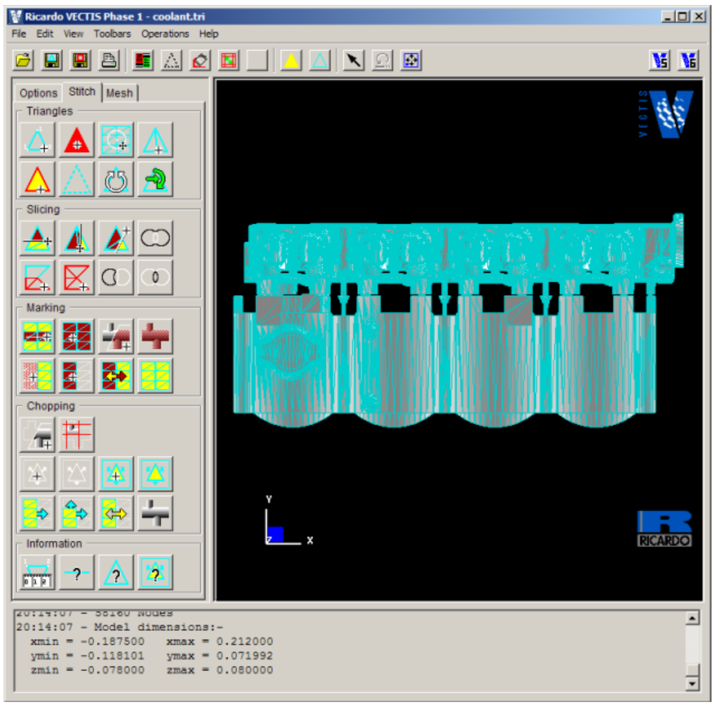
After copying the coolant.tri file to your working directory using the hyperlink at the top of this page it can be loaded into the VECTIS-MAX pre-processor Phase1 and should be displayed as shown by Figure 19.62.

When performing a VECTIS-MAX analysis, a closed geometry file of the analysis domain is required. For this tutorial this is provided in the form of the coolant.tri file. For future analyses the geometry can be repaired using the Phase1 repair tools. However in order to reduce the analysis preparation time it is more efficient to ensure that the STL file being exported from the CAD package is as complete and as closed as possible.

In particular for coolant analyses it is obviously crucial to get an accurate representation of the gasket holes. Therefore if the coolant jacket geometry from the engine block, cylinder head and gasket holes can be merged to create one solid in the CAD package and this then exported as an STL file the preparation time will be significantly reduced in Phase1.

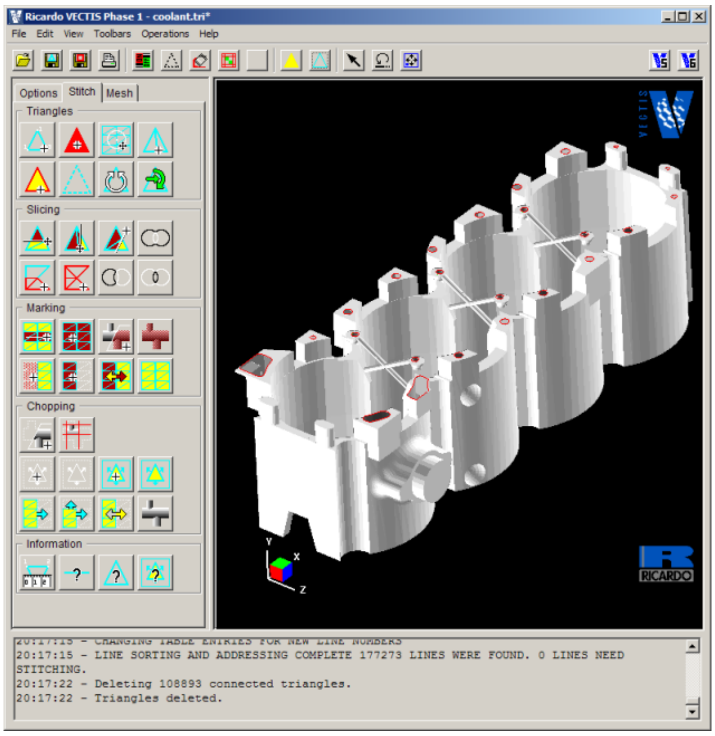
Figure 19.63 shows the gasket detail included in this coolant jacket example.

In this tutorial the geometry is fully stitched ready for meshing. The user can now enter the boundary identification section of Phase 1. The purpose of this section is to define different regions of the surface as different boundaries. In this example, the user needs to distinguish the different regions that will become the walls and inlet/outlet boundaries.


Ricardo VECTIS Phase 1 - coolant.tri
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Slicing
Marking
Chopping
Information
RICARDO

```
20:14:07 - 58160 Nodes
20:14:07 - Model dimensions:-
  xmin = -0.187500   xmax = 0.212000
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
```

Figure 19.62: The coolant jacket geometry when loaded into Phase1

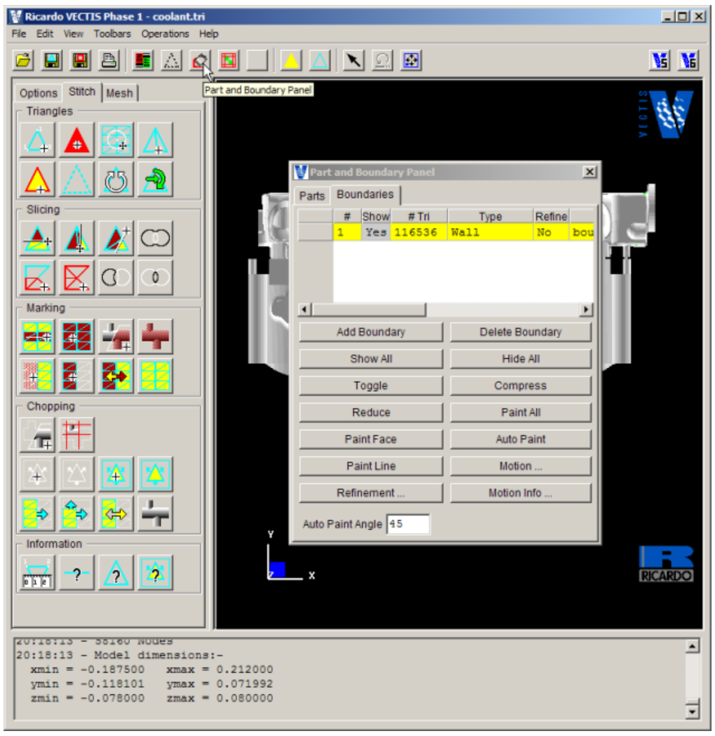

Ricardo VECTIS Phase 1 - coolant.tri*
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Slicing
Marking
Chopping
Information
RICARDO

```
20:17:15 - CHANGING TABLE ENTRIES FOR NEW LINE NUMBERS
20:17:15 - LINE SORTING AND ADDRESSING COMPLETE 177273 LINES WERE FOUND. 0 LINES NEED
STITCHING.
20:17:22 - Deleting 108893 connected triangles.
20:17:22 - Triangles deleted.
```

Figure 19.63: Gasket Hole Detail

19.3.3.1 Selecting Boundaries

Once the geometry has been loaded in to Phase1 the first action is to paint the inlet and outlet boundaries. Do this using the Operations > Boundary Painting tools.



#	Show	# Tri	Type	Refine	
1	Yes	116536	Wall	No	bou

```
20:18:13 - Model dimensions:-
  xmin = -0.187500    xmax = 0.212000
  ymin = -0.118101    ymax = 0.071992
  zmin = -0.078000    zmax = 0.080000
```

Figure 19.64: The Boundary Painting Panel

The inlet and outlet boundaries should be painted as universal boundaries 2 and 3 respectively as shown by figure 19.65 using the Paint Face tool. Also the type option for universal boundaries 2 and 3 should be set to Inlet/Outlet by right clicking the mouse on the row type entry for each boundary. The final setup is shown by Figure 4.

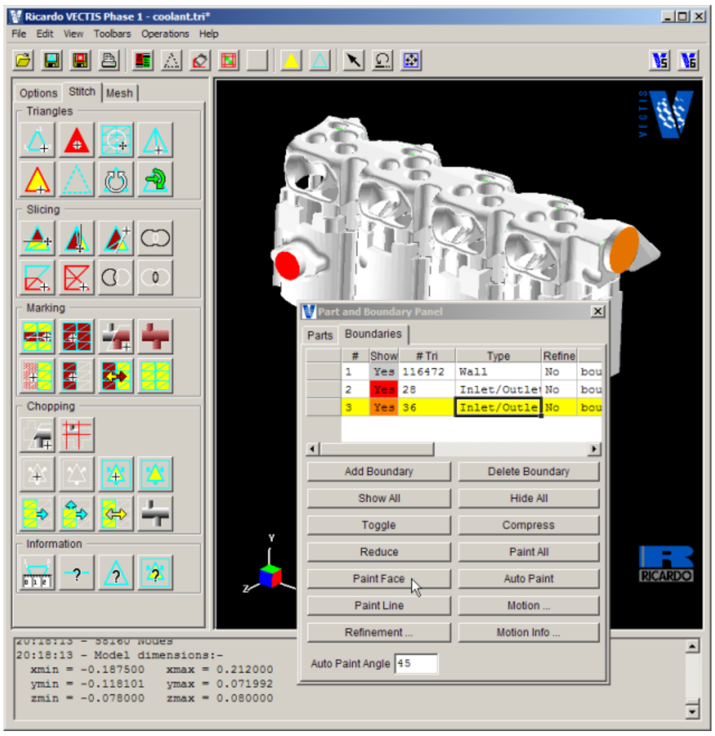
1. Use the 'Paint Face' tool to mark the Inlet/Outlet Boundaries

2. Click on the type box to toggle to the correct boundary type

After painting the boundaries save the geometry file as coolant.tri. The next stage is to setup the control mesh for the coolant jacket analysis. Activate the Mesh Setup toolbar as shown by Figure 19.66.

Phase1 automatically positions the two external red control lines around the geometry. In general it can be beneficial to re-position these lines closer the the model extents as shown by Figure 19.67. This should ensure that the aspect ratio of the cells adjacent to these lines is as good as possible. This can be done using the movement icon or by creating new red control lines and deleting the original red lines.

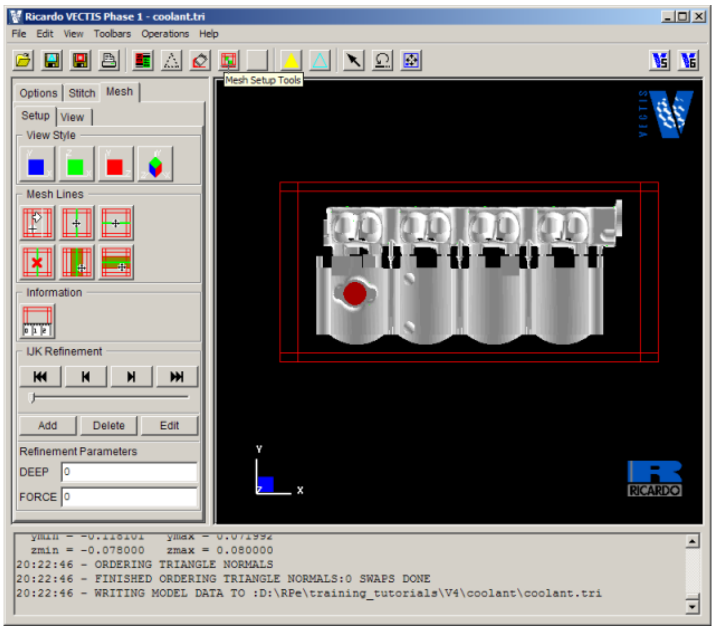
Use the mesh division lines to position the red control lines closer to the model extents and add the gasket control lines.



	#	Show	# Tri	Type	Refine	
	1	Yes	116472	Wall	No	bou
	2	Yes	28	Inlet/Outle	No	bou
	3	Yes	36	Inlet/Outle	No	bou

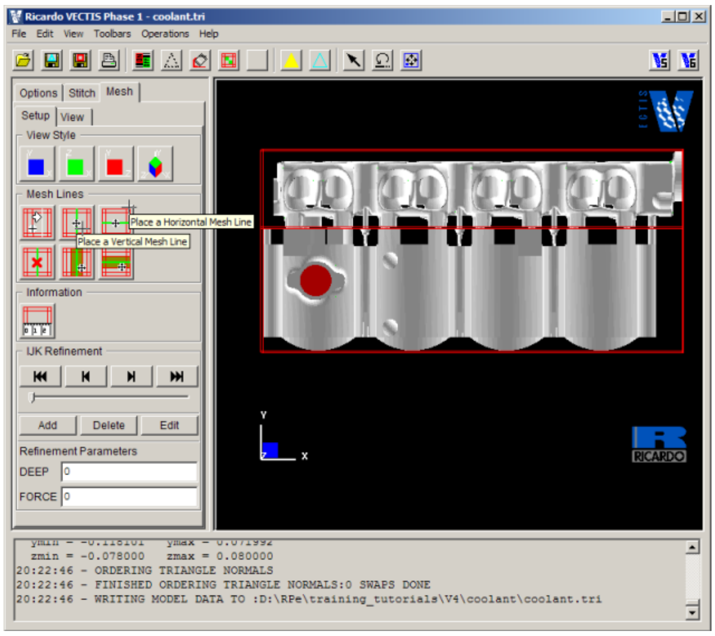
```
20:18:13 - 58160 Nodes
20:18:13 - Model dimensions:-
  xmin = -0.187500   xmax = 0.212000
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
```

Figure 19.65: The coolant jacket boundary setup



```
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
20:22:46 - ORDERING TRIANGLE NORMALS
20:22:46 - FINISHED ORDERING TRIANGLE NORMALS:0 SWAPS DONE
20:22:46 - WRITING MODEL DATA TO :D:\RPe\training_tutorials\V4\coolant\coolant.tri
```

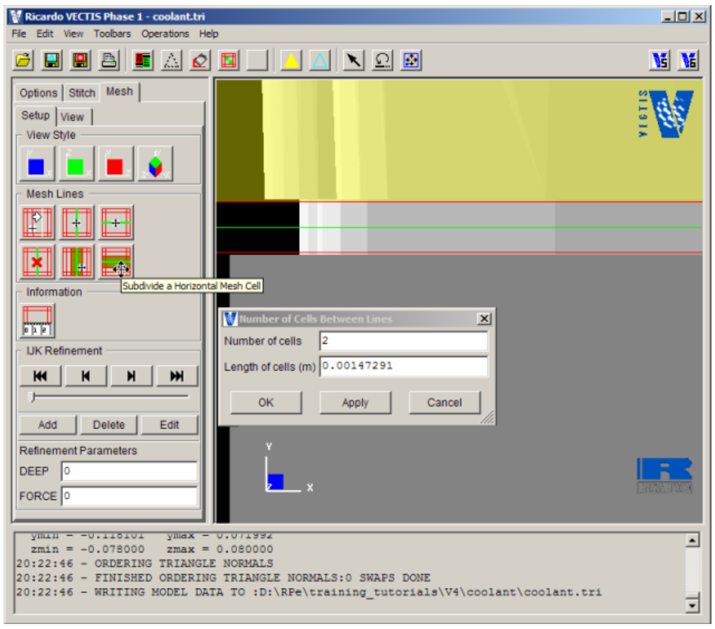
Figure 19.66: The mesh setup toolbar



```
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
20:22:46 - ORDERING TRIANGLE NORMALS
20:22:46 - FINISHED ORDERING TRIANGLE NORMALS:0 SWAPS DONE
20:22:46 - WRITING MODEL DATA TO :D:\RPe\training_tutorials\V4\coolant\coolant.tri
```

Figure 19.67: Re-positioning the external red control lines

For coolant analyses the geometry scale of the main coolant jacket to the gasket geometry is usually significantly different. Therefore it is good practice to use red control lines to control the mesh cell sizes which are located in the gasket. Create two new mesh lines positioned at the height extents of the gasket. Also define the number of intervals between these lines as 2. This is shown by Figure 19.68.



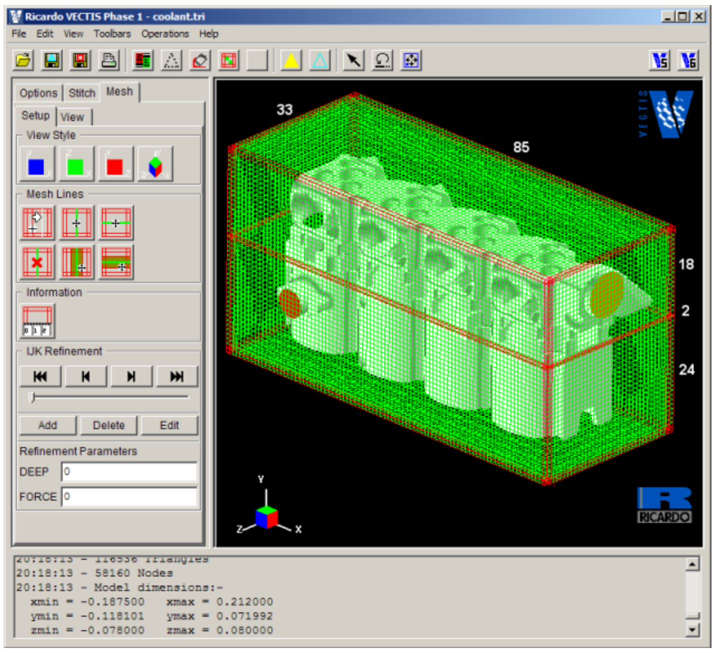
Number of cells: 2
Length of cells (m): 0.00147291

```
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
20:22:46 - ORDERING TRIANGLE NORMALS
20:22:46 - FINISHED ORDERING TRIANGLE NORMALS:0 SWAPS DONE
20:22:46 - WRITING MODEL DATA TO :D:\RPe\training_tutorials\V4\coolant\coolant.tri
```

Figure 19.68: Using meshlines to control the cell sizes located in the gasket region

The divisions for the other sections between the red control lines can now be defined. For this example the computational control mesh cell size will be set as 5mm x 5mm x 5mm. For future analyses it is up to the user to determine what size cells to use. Typical cell sizes are 2mm to 4mm.

The spacing distance between the red lines can be determined by using the measurement icon in the Mesh Setup Toolbar - this is the icon at the bottom. Click this icon and then right click on the two parallel red control lines that you wish to measure between. The distance is then displayed in the Phase1 window. This distance can be divided by the required mesh cell size to give the required number of divisions. This number of intervals can then be set up for each interval. The number of divisions for each section for this example mesh are shown by Figure 19.69.



```
20:18:13 - 116536 Triangles
20:18:13 - 58160 Nodes
20:18:13 - Model dimensions:-
  xmin = -0.187500   xmax = 0.212000
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
```

Figure 19.69: Example control mesh number of sub-divisions to give approximately 5mm size cells

The final step for the mesh file setup is to set the boundary refinement level.

This is done through the Operations > Mesh File Setup panel. The boundary refinement entry should be set to 1. This means that every cell at the surface can be refined by $2^1 = 2$ in each co-ordinate direction. The setup for this is shown by Figure 19.70.

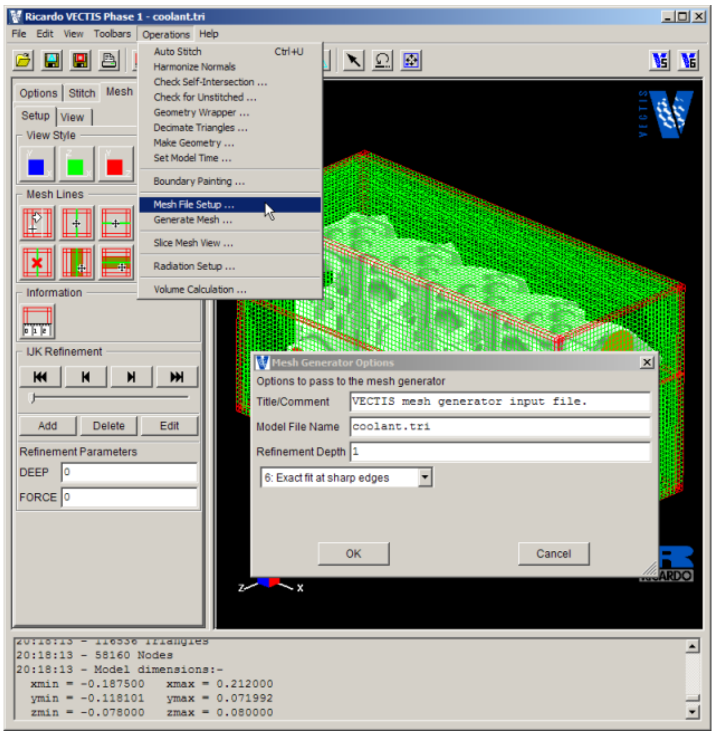
Save the mesh file as coolant.mesh. The mesh setup is now complete.

19.3.4 Mesh Generation

In order to generate the computational mesh the .tri and .mesh files produced by Phase1 are needed. The mesher, vmesh, can either be started from the command line or within R-desk. In this case vmesh will be run from the command line. In a terminal or dos command window type the following command:

```
%vmesh coolant.mesh
```

The mesher will write output to the screen. This screen output is also written to a file, in this case coolant.OUT. If this file is opened (or if the mesher was run from the command line) you can



Refinement Parameters

DEEP 0

FORCE 0

Options to pass to the mesh generator

Title/Comment: VECTIS mesh generator input file.

Model File Name: coolant.tri

Refinement Depth: 1

6: Exact fit at sharp edges

```
20:18:13 - 116536 Triangles
20:18:13 - 58160 Nodes
20:18:13 - Model dimensions:-
  xmin = -0.187500   xmax = 0.212000
  ymin = -0.118101   ymax = 0.071992
  zmin = -0.078000   zmax = 0.080000
```

Figure 19.70: Using meshlines to control the cell sizes located in the gasket region

see a summary of the mesh generation written at the end of mesh generation. In this case it can be seen that two warnings are produced. These are non fatal and the .GRD computation mesh file is written.

The summary output is shown below:

```
-----------------------------------------------------------------
STATISTICAL DATA OF GENERATED MESH
Number of generated cells: 64181 (boundary: 48024 ; internal: 16157)
--- Patching method ---
Number of cells processed by Marching Cubes: 36802 (72.48 %)
Number of cells processed by Exact Fit: 9165 (18.05 %)
Number of volumes broken by Cell Splitting: 2722 (5.36 % of boundary cells)
  [651 attempts to split a volume failed (19.30 % of all attempts)
   274 attempts gave incorrect number of volumes
   377 attempts gave volumes with too low quality, so the undo was applied]
--- Cell quality ---
Number of correct boundary cells: 48714 (95.94 %)
NO PROBLEMS with negative volumes
Number of cells with problems of gaps: 2 (0.00 %)
There were 2348 small volumes, they are deleted now (4.62 %)
Number of cells which had to be deactivated: 2057 (4.05 %)
Total mesh volume: 1.75590e-003 u3
[ = 1.09865e-003 u3 (16157 inner cells) + 6.57243e-004 u3 (48883 boundary cells)]
-----------------------------------------------------------------

WRITING THE MESH FILE
- successfully finished

Total time elapsed: 54 seconds

DURING THE MESHING, 2 OF WARNING(S) OR NON-FATAL ERRORS APPEARED:

1 X
ERROR #1506: Tying patches routine: cannot move the required node
             in box 37712
  in these IJK positions:
    42 31 3
1 X
ERROR #1505: Tying patches routine: cannot break the required edge
             in box 53463
  in these IJK positions:
    58 31 3


SUCCESSFULLY DONE, BUT WARNINGS APPEARED.
CHECK MESH QUALITY.
```

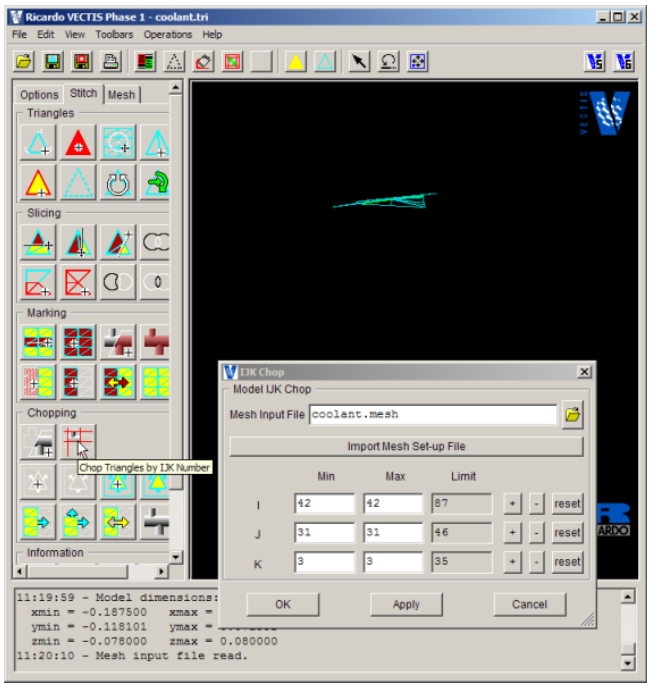
Normally the solver will run successfully with these non fatal error however occasionally these problematic areas can hinder convergence. The mesher output indicates the IJK regions where the errors occur. These can be visualised in Phase1 and any obvious problematic geometry can be rectified by the user.

Load the coolant.tri and coolant.mesh file into Phase1. The tool 'chop triangles by IJK number' quickly allows the problematic geometry to be visualised. Enter in the location of the first IJK region (42, 31, 3). 19.71.

Switch on the visualisation of surfaces and lines to make things easier to see. (In the option tab check on 'show surfaces' and set 'outline colours' to mono.) It may be necessary to 'grow' the chopped area to visualise exactly what the geometry looks like. 19.72.

It can be seen that there is a very thin triangle which creates an 'overlap' (highlighted in green). The easiest way to rectify this is to delete the triangle and its neighbours and create new triangles with a better aspect ration. 19.73.

Use the 'chop triangles by IJK number' tool again to visualise the region (58, 31, 3). A small imperfection can be seen in the surface.

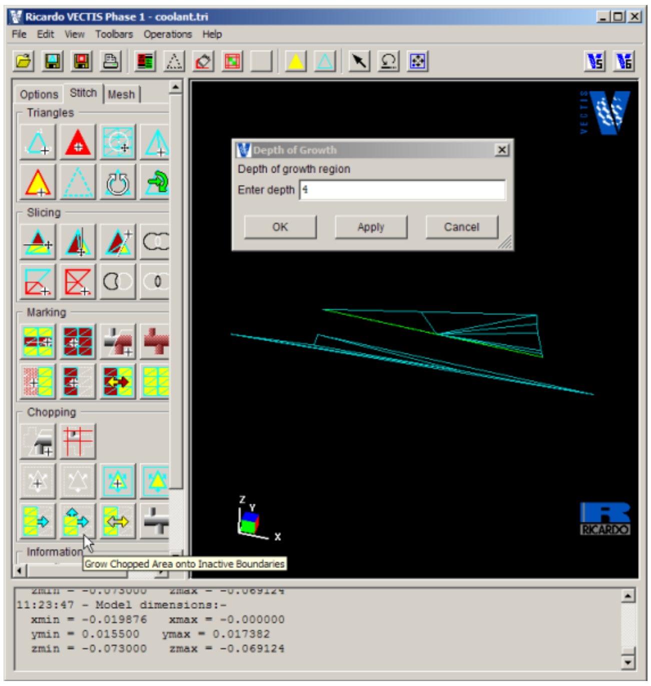


Mesh Input File coolant.mesh

	Min	Max	Limit
I	42	42	87
J	31	31	46
K	3	3	35

```
11:19:59 - Model dimensions:
  xmin = -0.187500    xmax =
  ymin = -0.118101    ymax =
  zmin = -0.078000    zmax = 0.080000
11:20:10 - Mesh input file read.
```

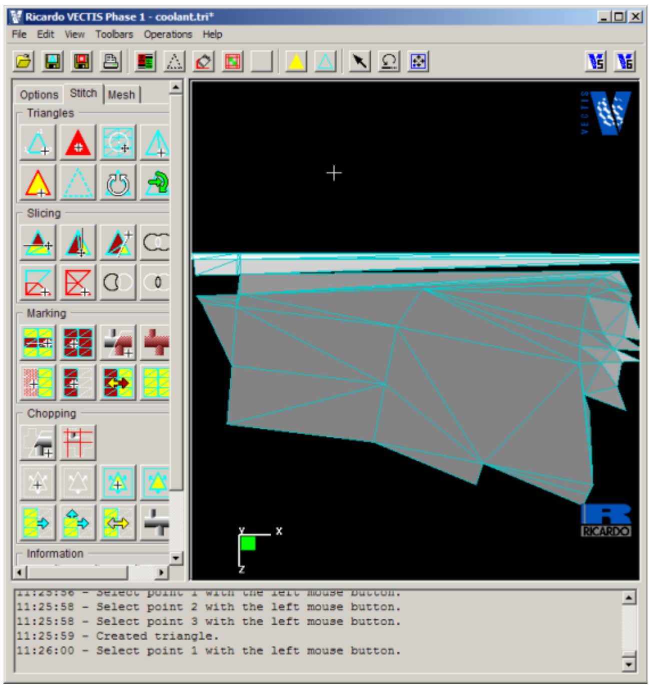
Figure 19.71: Chopping the geometry by IJK region


Ricardo VECTIS Phase 1 - coolant.tri
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Slicing
Marking
Chopping
Information
Grow Chopped Area onto Inactive Boundaries
Depth of Growth
Depth of growth region
OK
Apply
Cancel

Enter depth 4

```
  zmin = -0.073000    zmax = -0.069124
11:23:47 - Model dimensions:-
  xmin = -0.019876    xmax = -0.000000
  ymin = 0.015500    ymax = 0.017382
  zmin = -0.073000    zmax = -0.069124
```

Figure 19.72: Growing the chopped geometry


Ricardo VECTIS Phase 1 - coolant.tri*
File Edit View Toolbars Operations Help
Options Stitch Mesh
Triangles
Slicing
Marking
Chopping
Information

```
11:25:58 - Select point 1 with the left mouse button.
11:25:58 - Select point 2 with the left mouse button.
11:25:58 - Select point 3 with the left mouse button.
11:25:59 - Created triangle.
11:26:00 - Select point 1 with the left mouse button.
```

Figure 19.73: Overlapping geometry region after repair

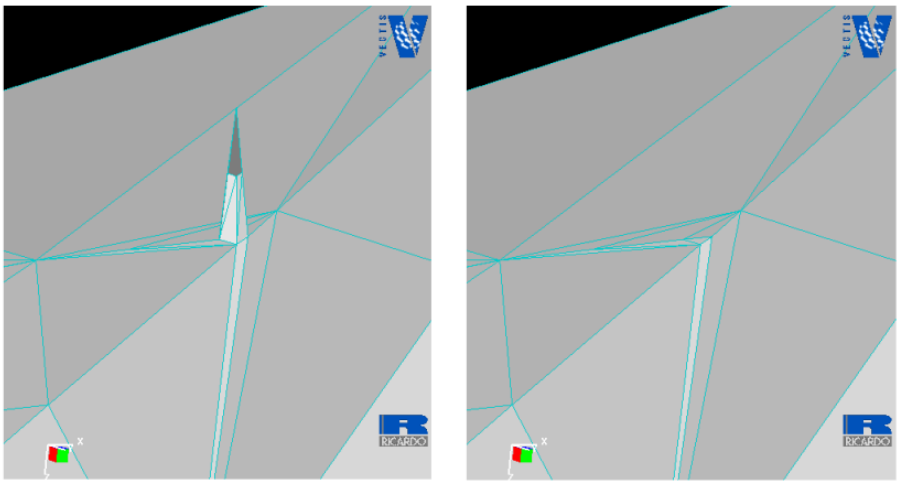


Figure 19.74: Problematic region in the geometry

Using the delete triangle and cap hole tools this imperfection can be removed. 19.74

```
-----------------------------------------------------------------
STATISTICAL DATA OF GENERATED MESH
Number of generated cells: 63113 (boundary: 46967 ; internal: 16146)
--- Patching method ---
Number of cells processed by Marching Cubes: 36534 (73.85 %)
Number of cells processed by Exact Fit: 8424 (17.03 %)
Number of volumes broken by Cell Splitting: 2503 (5.06 % of boundary cells)
  [618 attempts to split a volume failed (19.80 % of all attempts)
   262 attempts gave incorrect number of volumes
   356 attempts gave volumes with too low quality, so the undo was applied]
--- Cell quality ---
Number of correct boundary cells: 47461 (95.93 %)
NO PROBLEMS with negative volumes
Number of cells with problems of gaps: 2 (0.00 %)
There were 2281 small volumes, they are deleted now (4.61 %)
Number of cells which had to be deactivated: 2009 (4.06 %)
Total mesh volume: 1.75013e-003 u3
[ = 1.09839e-003 u3 (16146 inner cells) + 6.51745e-004 u3 (47634 boundary cells)]
-----------------------------------------------------------------


WRITING THE MESH FILE
- successfully finished

Total time elapsed: 52 seconds

SUCCESSFULLY DONE
```

Now that the computational mesh has been generated we can move onto the next stage, setting up the input file for the solver.

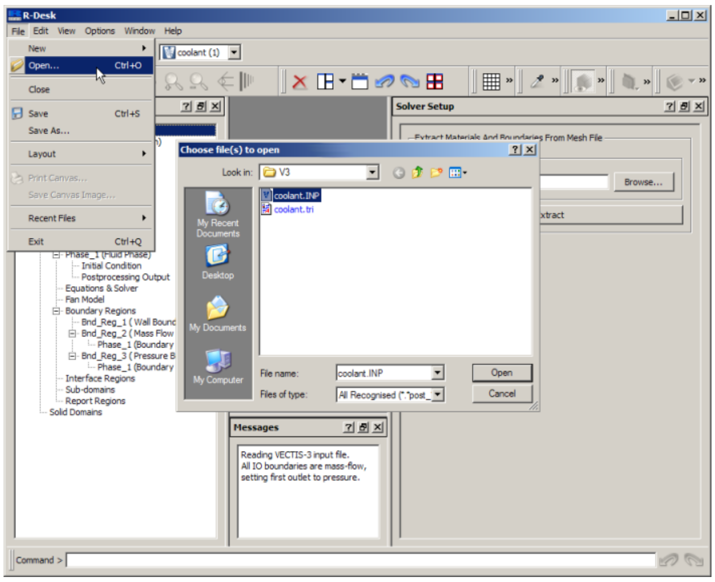
19.3.5 Solver Setup

Coolant analyses for gasket optimisation and general coolant jacket development are usually steady state since both the geometry and flow properties are constant with time. Therefore for this example the analysis will be setup to use the VECTIS-MAX steady state solver. This is an example analysis which is intended to run quickly and so the end time will be set to 500 iterations.

Since the analysis is steady state the only Postprocessing file required is at the analysis end time or when a converged solution has been achieved. In this instance an existing set-up from a VECTIS 3 calculation will be imported to set-up the majority of the input.

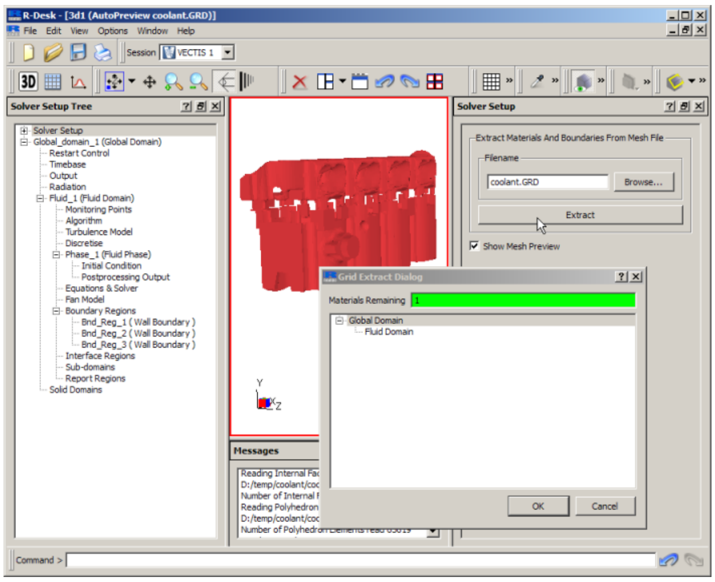
Open *R–Desk* , select 'File' > 'Open' and browse to the coolant_V3.INP file. This will import the file and open a new VECTIS session in rdesk. The fluid properties, timebase settings and boundary condition specifications should be read in from the old input file. One thing to note is that in the VECTIS-MAX solver it is recommended to have at least one non mass-flow boundary to ensure the the boundary conditions are posed correctly. In the original VECTIS 3 set-up both the boundaries were specified as mass flow. As such when the input file is imported, the first outlet mass flow boundary is changed to a pressure boundary; as indicated in the messages area. **??**

Additionally the GRD file can be imported. This will allow the boundary definitions imported from the existing V3 input file to be verified against the boundaries defined in the GRD file. Click on 'Solver Setup' found at the top of the Solver Setup Tree and select the coolant.GRD file in the file browser. Check mesh preview and select import . (Figure 19.76). Now when the boundary regions are selected int he Solver Setup Tree they should be highlighted in the 3d canvas.



Reading VECTIS-3 input file.
All IO boundaries are mass-flow,
setting first outlet to pressure.

Figure 19.75: Importing VECTIS 3 solver setup



Reading Internal Fac
D:/temp/coolant/coo
Number of Internal F
Reading Polyhedron
D:/temp/coolant/coo
Number of Polyhedron Elements read 65019

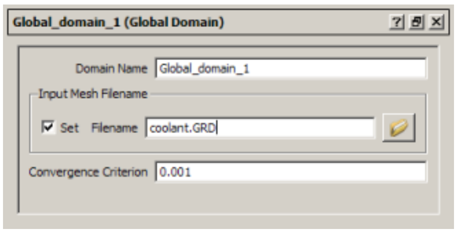
Figure 19.76: Importing the computational grid into the solver setup

Now we will run quickly through the Solver Setup Tree . More detailed information on all of the panels in the tree can be found in the manual or in previous tutorials.

19.3.5.1 Global Domain

Click on the global domain entry in the Solver Setup Tree . The global domain panel allows general simulation parameters to be set. (Figure 19.77)

Input Mesh Filename The solver will default to using a computational grid file with the name projectname.GRD (where the projectname is taken as the .inp file prefix). A different grid file can



Global_domain_1 (Global Domain)

Domain Name: Global_domain_1

Input Mesh Filename

☑ Set Filename: coolant.GRD

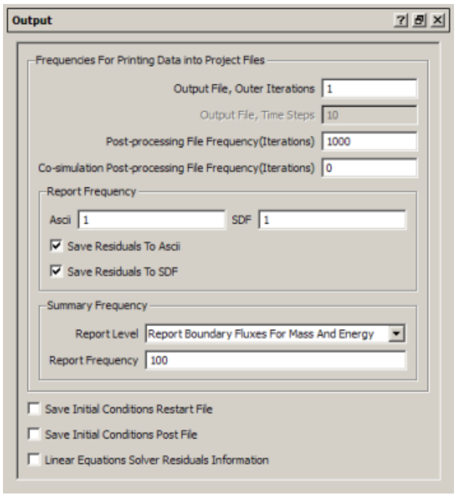
Convergence Criterion: 0.001

Figure 19.77: The Global Domain Panel

be specified here. Again here we will use the .GRD file that was previously generated, normally coolant.GRD.

Convergence Criterion This is the criteria used to determine whether the simulation has converged. In general, as the complex flow within the coolant circuit is never completely steady and tends to fluctuate in localised regions, the simulation will have higher residual levels than other types of calculations. As such a more relaxed criteria can be used in order to prevent excessive computation after global levels (such as mass flow through gasket holes) have converged. In this case a criterion of 1e-4 will be specified.

Timebase Click on the Timebase entry in the Solver Setup Tree . This is were the timebase/iteration settings are input. These should have been read in from the VECTIS 3 input file. The should be 'Steady State' with 500 Iterations.



Output

Frequencies For Printing Data into Project Files

Output File, Outer Iterations: 1

Output File, Time Steps: 10

Post-processing File Frequency(Iterations): 1000

Co-simulation Post-processing File Frequency(Iterations): 0

Report Frequency

Ascii: 1 SDF: 1

☑ Save Residuals To Ascii

☑ Save Residuals To SDF

Summary Frequency

Report Level: Report Boundary Fluxes For Mass And Energy

Report Frequency: 100

☐ Save Initial Conditions Restart File

☐ Save Initial Conditions Post File

☐ Linear Equations Solver Residuals Information

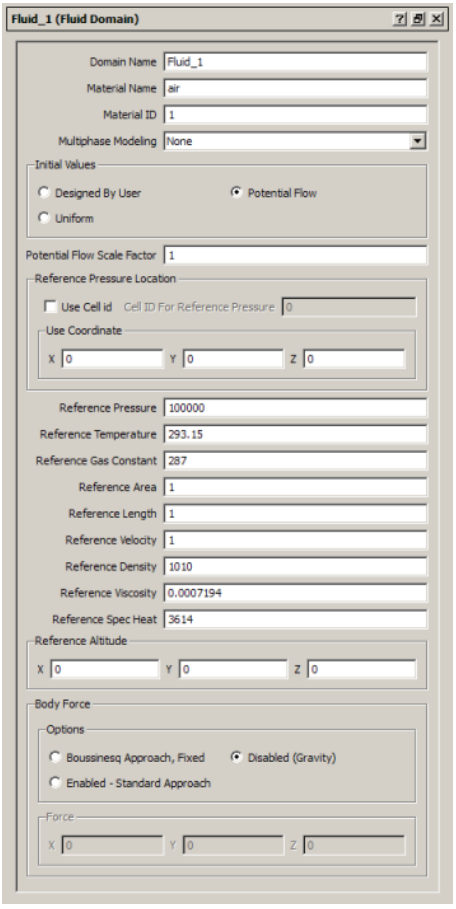
Figure 19.78: Output Options

Output Here, set the postprocessing frequency to a high value as data is only required at the end of the simulation. Also set the 'Ascii' report frequency to 1 so that the files are available for use with live update . Additionally check the residual data options so that the residual values are written to ascii files and the .rep sdf file. (Figure 19.78)

Fluid Domain In this case the calculation will consist of one single-phase fluid domain which is liquid to represent the coolant.

The potential flow initialisation will be used to set-up the initial conditions.

The reference data for density, viscosity and specific heat should be changed to match the constant values found in the Fluid Phase setup which was imported from the old input file.



Fluid_1 (Fluid Domain)

Domain Name	Fluid_1
Material Name	air
Material ID	1
Multiphase Modeling	None

Initial Values: Designed By User, [✓ Potential Flow], Uniform

Potential Flow Scale Factor	1

Reference Pressure Location: Use Cell id (unchecked), Cell ID For Reference Pressure 0

Use Coordinate: X 0, Y 0, Z 0

Reference Pressure	100000
Reference Temperature	293.15
Reference Gas Constant	287
Reference Area	1
Reference Length	1
Reference Velocity	1
Reference Density	1010
Reference Viscosity	0.0007194
Reference Spec Heat	3614

Reference Altitude: X 0, Y 0, Z 0

Body Force

Options: Boussinesq Approach, Fixed, [✓ Disabled (Gravity)], Enabled - Standard Approach

Force: X 0, Y 0, Z 0

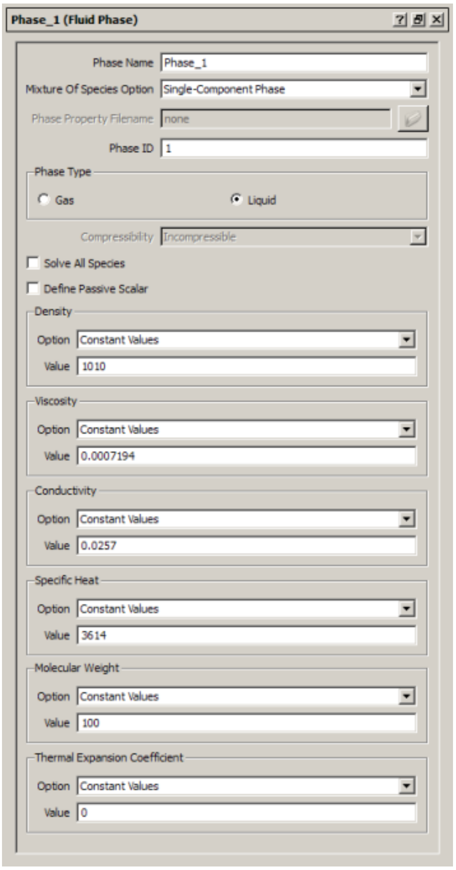
Figure 19.79: The Fluid Domain Panel

The remaining fluid panels (algorithm, turbulence model, equations & solver and discretise) will be left at their default values.

Fluid Phase The fluid phase panel contains data for each fluid phase. In this case the properties for the single phase should have been imported from the old input file. Figure 19.80

Boundary Regions Each of the boundary regions specifications should have been imported from the old input file. As noted previously the second mass flow boundary defined in the original input file has been changed to a pressure boundary. In these cases it is important to verify that the conditions being simulated still correspond to the physical boundary conditions.

The only modification that should be necessary for the boundary regions is to check the 'Boundary



Phase_1 (Fluid Phase)

Phase Name	Phase_1
Mixture Of Species Option	Single-Component Phase
Phase Property Filename	none
Phase ID	1

Phase Type: ( ) Gas [✓ Liquid]

Compressibility	Incompressible

☐ Solve All Species

☐ Define Passive Scalar

	Option	Value
Density	Constant Values	1010
Viscosity	Constant Values	0.0007194
Conductivity	Constant Values	0.0257
Specific Heat	Constant Values	3614
Molecular Weight	Constant Values	100
Thermal Expansion Coefficient	Constant Values	0

Figure 19.80: Fluid Phase Options

Report' option for the inlet and outlet boundaries. See Figure 19.81.

The following boundary conditions will be used.

– Boundary 1 will be a wall with a fixed temperature of 373 deg K

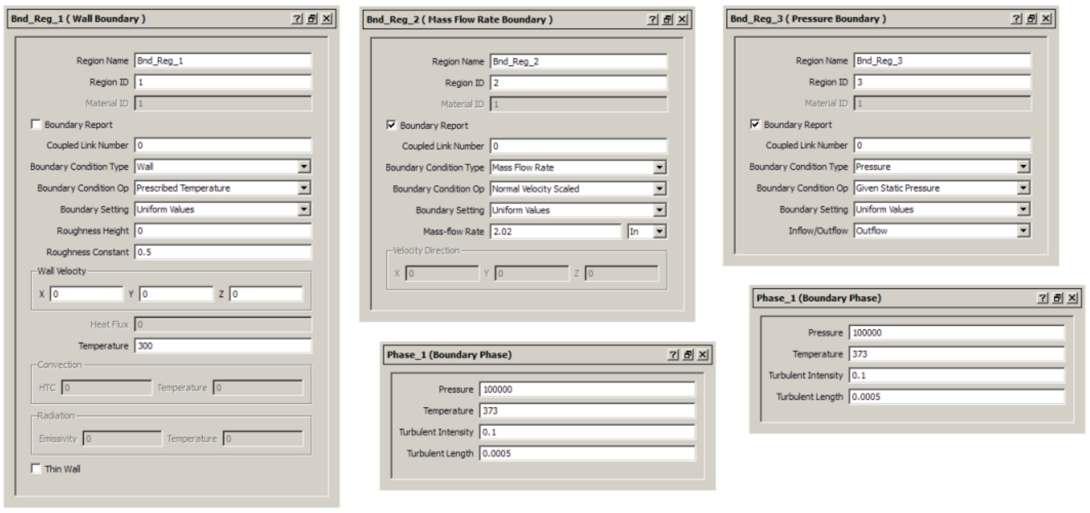
– Boundary 2 will be a specified mass flow boundary with a rate of 2.02 kg/s with a temperature of 373 deg K

– Boundary 3 will be a static pressure boundary set to 100000 Pa, outflow and with a temperature of 373 deg K

The input file should now be complete for the analysis. Save the file as coolant.inp ('file' > 'save as'). On important thing to note is that VECTIS-MAX uses a lower case .inp extension for the solver input file.

19.3.6 Grid Preparation

Vpre prepares the computational grid for use in the solver. It can be started either from a cmd window or from *R-desk*. In this case we shall run it from a command window. Type the following command to run the tivpre mesh pre-processor.


Bnd_Reg_1 (Wall Boundary)

Region Name	Bnd_Reg_1
Region ID	1
Material ID	1
Boundary Report	☐
Coupled Link Number	0
Boundary Condition Type	Wall
Boundary Condition Op	Prescribed Temperature
Boundary Setting	Uniform Values
Roughness Height	0
Roughness Constant	0.5
Wall Velocity X	0
Wall Velocity Y	0
Wall Velocity Z	0
Heat Flux	0
Temperature	300
Convection HTC	0
Convection Temperature	0
Radiation Emissivity	0
Radiation Temperature	0
Thin Wall	☐

Bnd_Reg_2 (Mass Flow Rate Boundary)

Region Name	Bnd_Reg_2
Region ID	2
Material ID	1
Boundary Report	✓
Coupled Link Number	0
Boundary Condition Type	Mass Flow Rate
Boundary Condition Op	Normal Velocity Scaled
Boundary Setting	Uniform Values
Mass-flow Rate	2.02 In
Velocity Direction X	0
Velocity Direction Y	0
Velocity Direction Z	0

Phase_1 (Boundary Phase)

Pressure	100000
Temperature	373
Turbulent Intensity	0.1
Turbulent Length	0.0005

Bnd_Reg_3 (Pressure Boundary)

Region Name	Bnd_Reg_3
Region ID	3
Material ID	1
Boundary Report	✓
Coupled Link Number	0
Boundary Condition Type	Pressure
Boundary Condition Op	Given Static Pressure
Boundary Setting	Uniform Values
Inflow/Outflow	Outflow

Phase_1 (Boundary Phase)

Pressure	100000
Temperature	373
Turbulent Intensity	0.1
Turbulent Length	0.0005

Figure 19.81: Boundary region settings

```
vpre coolant.GRD
```

vpre is also be used to re-partition the mesh for parallel calculations using the -np flag.

19.3.7 Running the solver

Now that the .inp file has been created and the grid file has been processed the solver can be run.

Again this will be run from the cmd window by typing the following:

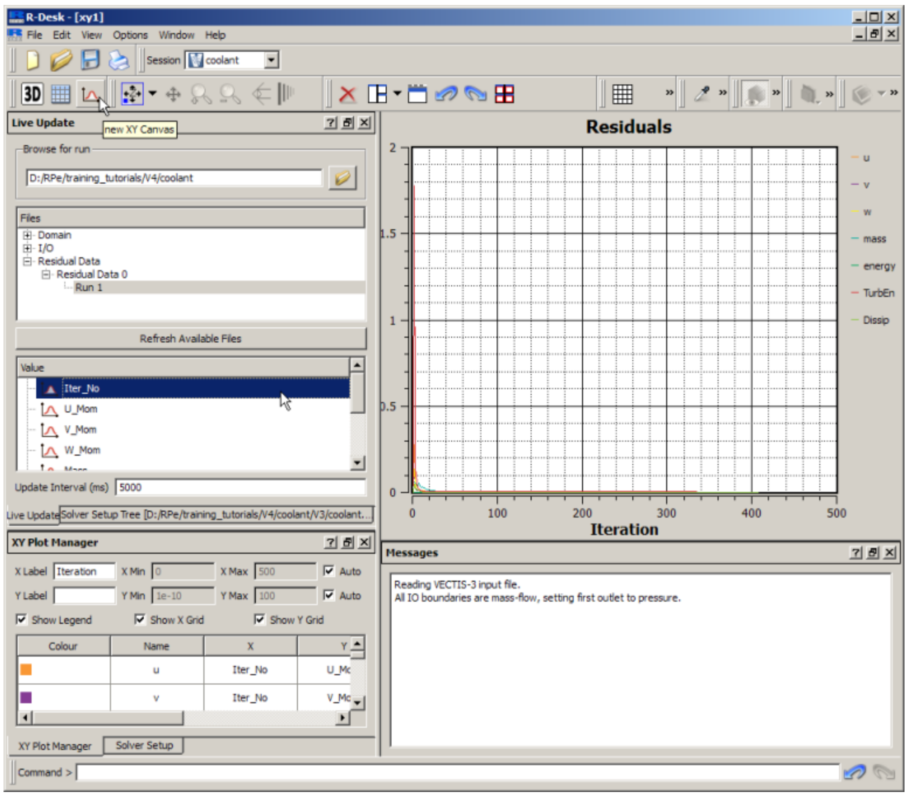
```
vsolve port.inp
```

The solver will now start to run.

19.3.8 Live Update

Using the live update is a utility in R-Desk allows a simulation to be monitored whilst it is running.

Within the VECTIS session in R-Desk open the Live Update and XY plot manager panels if they are not already open. Then open an xy canvas using the new XY canvas button. In the Live Update panel, browse to the directory where the simulation is running. The available data files are presented in the 'Files' window. The data files available are determined by the ascii files selected in the reporting section of the 19.3.5.1 in the solver setup tree. Select the residual data entry for the appropriate run.The data available in the file is shown in the 'Value' window. Drag the data from this panel in pairs (Iter_Num Vs Residual) to the XY canvas. It will then be plotted. Each curve on the plot will appear in the XY plot manager . In this panel the properties of the individual curves and of the XY plot can be modified.



Browse for run: D:/RPe/training_tutorials/V4/coolant

Update Interval (ms): 5000

Live Update | Solver Setup Tree [D:/RPe/training_tutorials/V4/coolant/V3/coolant...

X Label: Iteration X Min: 0 X Max: 500 [✓ Auto]
Y Label: Y Min: 1e-10 Y Max: 100 [✓ Auto]
[✓ Show Legend] [✓ Show X Grid] [✓ Show Y Grid]

Colour	Name	X	Y
🟧	u	Iter_No	U_Mc
🟪	v	Iter_No	V_Mc

Reading VECTIS-3 input file.
All IO boundaries are mass-flow, setting first outlet to pressure.

Figure 19.82: Live Update

Additional the scale can be changed to a logarithmic type by right clicking the relevant scale, Figure 19.83. Looking at the residual values it can be seen that the majority have stablised after approximately 100 iterations.

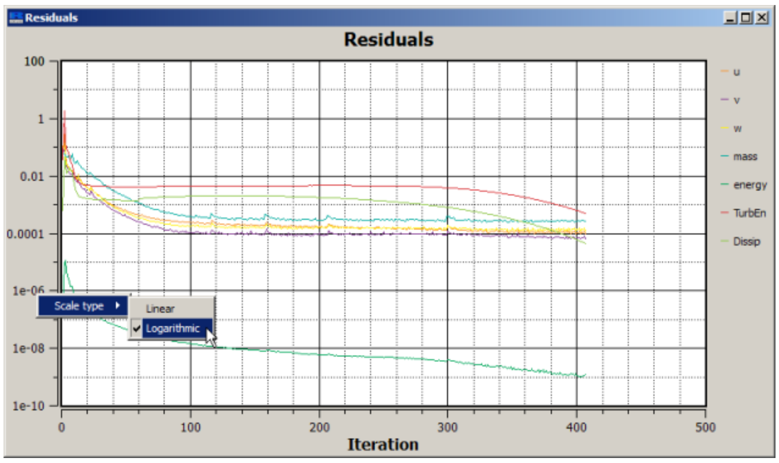


Figure 19.83: The residual Values for the coolant simulation

Additionally plotting the massflow rate on the outlet pressure boundary shows the the convergence of the simulations, Figure 19.84

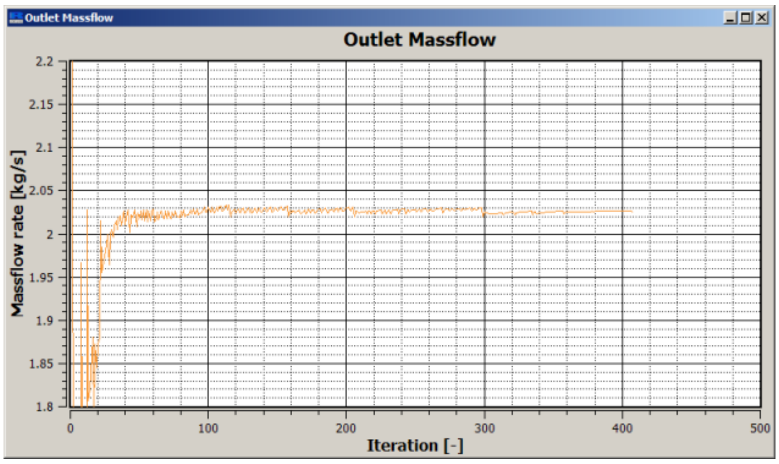


Figure 19.84: Massflow at the outlet pressure boundary

19.3.9 Post-processing

The simulation should reach the convergence criteria after around 400 iterations at which point the simulation will stop and write out post and restart files. Open the post file in R-Desk using the *'File'* > *'Open'* menu option. It will appear in the plot tree . By default the plot will be added to the active 3D canvas. Note: this behaviour can be changed in the preferences panel.

The attributes of the plot can be modified in the Plot Properties panel

Different data can be shown on the plot by selecting it in the Data panel as shown in Figure 19.85

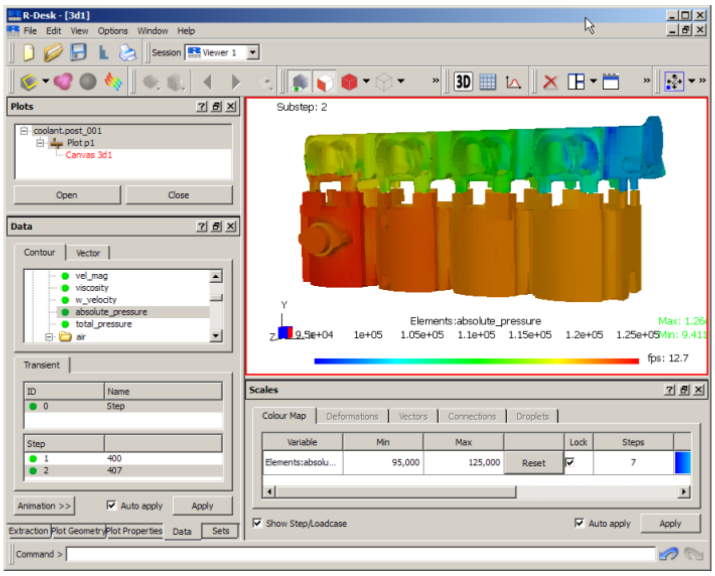


Figure 19.85: Pressure variation at the model surface

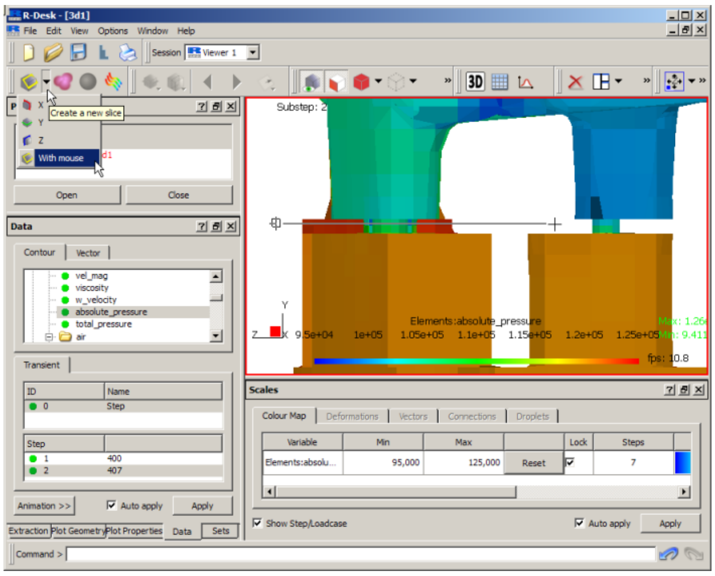
19.3.9.1 Data extraction

Typically the purpose of performing a coolant flow analysis is to determine two main things.

- ☐ System pressure drop across the coolant circuit
- ☐ Mass flow split through the gasket holes

The most straightforward way to determine the system pressure drop from inlet to outlet is to view the ascii .io files or the binary .rep report file. This will show the pressures and mass flows at all the boundaries. Alternatively if the pressure drop is desired from locations away from the boundaries then slice plots can be created in the locations of interest and the static pressure plotted on the slice. Then right-click on the slice plot in the plot tree and select 'Calculate' > 'Mean (Weighted)' > 'Area_Mean'. This will report the average pressure for the slice plot.

The mass flow split through the gasket holes can be determined in a couple of ways. Arbitrary surfaces may be used to output this data as the simulation is running. See the Port flow tutorial for instruction on using arbitrary surfaces. Additionally the mass flow split can also be extracted during post-processing in *rdesk*.



R-Desk - [3d1]
File Edit View Options Window Help
Session Viewer 1
Create a new slice
With mouse
Open
Close
Data
Contour Vector
vel_mag
viscosity
w_velocity
absolute_pressure
total_pressure
air
Transient
ID Name
0 Step
Step
1 400
2 407
Animation >>
Auto apply
Apply
Extraction Plot Geometry Plot Properties Data Sets
Command >
Substep: 2
Elements:absolute_pressure
9.5e+04 1e+05 1.05e+05 1.1e+05 1.15e+05 1.2e+05 1.25e+05
fps: 10.8
Scales
Colour Map Deformations Vectors Connections Droplets
Variable Min Max Lock Steps
Elements:absolu... 95,000 125,000 Reset 7
Show Step/Loadcase
Auto apply
Apply

Figure 19.86: Using the mouse to create a slice

To do this first create a plane through the gasket holes. In this case the plane will be defined using mouse input. Firstly orientate the model to view from one end so that a line can be drawn to define a plane through all the gasket holes, see figure 19.86. (Note: It may be beneficial to turn off perspective viewing when defining the plane.) Then click on the slice tool and select 'With mouse'. Next left-click once in the canvas to start the mouse definition.

Next define the plane by left-clicking two points in the canvas. A slice plot will appear in the plot tree . Note: The mouse mappings set in the preferences for 'pick' will determine the mouse (and keyboard) input needed for defining the slice.

If the slice plot is not already plotted in the 3d canvas drag the slice plot from the plot tree to the canvas. Remove the main model plot from the canvas by right-clicking the corresponding canvas name under the main model plot in the plot tree and selecting remove.

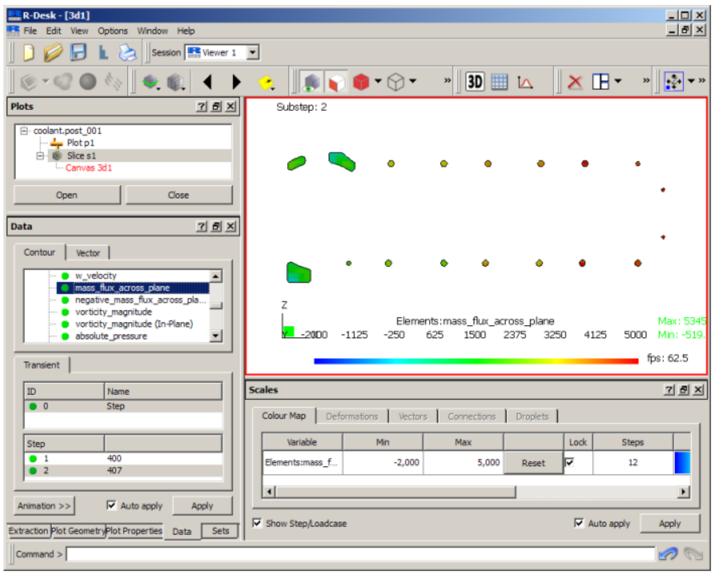


Figure 19.87: Visualising a slice through the gasket holes

Select the slice plot in the plot tree and then in the data panel select the contour data mass_flux_-across_plane, This will show the mass flow distribution through the gasket. To find the mass flow through each of the gasket hole passages the extract flux tool can be used. Right-click on the slice plot in the plot tree then select 'Extract' > 'Flux and Torque', Figure 19.88.

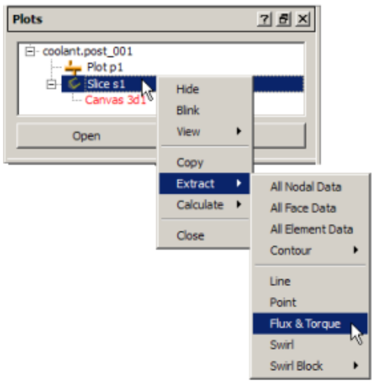


Figure 19.88: Extract Flux Options

This will bring up the extraction panel. Use the polygon option to click around the gasket holes. Normally a left-click in the canvas is required to define the polygon points. Figure 19.89

The mass flux through the selected region will then be written to the information panel. Figure 19.90

More information on the general post-processing capabilities can be found in the R-Desk section of the help.

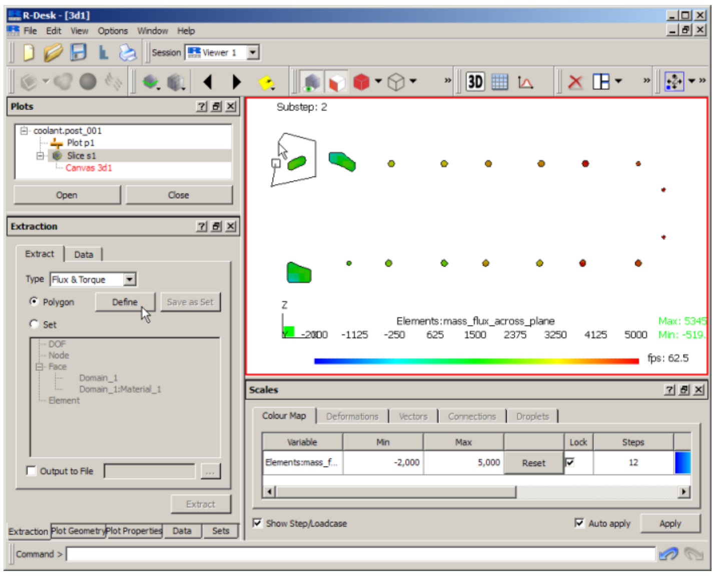
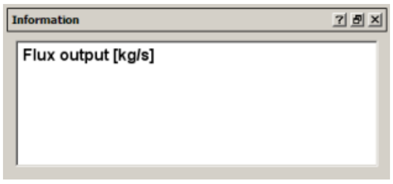


Figure 19.89: Using a polygon to define the region to extract flux data from



```
Flux output [kg/s]
```

Figure 19.90: Flux data written to information panel

19.4 Importing Third Party Meshes

Get the necessary files for the importing third party meshes tutorial

`http://www.software.ricardo.com/support/tutorials/vectismax/TutorialFiles`
`MeshImport`

19.4.1 Introduction

VECTIS-MAX allows any computational mesh to be solved, either the mesh files produced by the VECTIS mesher or third party meshes. The purpose of this tutorial is to illustrate importing external third party computational meshes into VECTIS-MAX.

The following items will be covered in this tutorial

- ☐ Converting external mesh format to native VECTIS-MAX .GRD file
- ☐ Defining boundary regions to a computational mesh using the set functionality of R-Desk
- ☐ Importing final computational grid into the soler setup

Currently supported mesh files types for import are:

1. I-Deas unv files. (here node sets can be used to identify the boundary regions)
2. CGNS files (Boundary regions should be imported from CGNS files)
3. CEDRE files (currently these require a .ccm extension, however these are typically exported as .ngeom files)
4. Spider .flma files

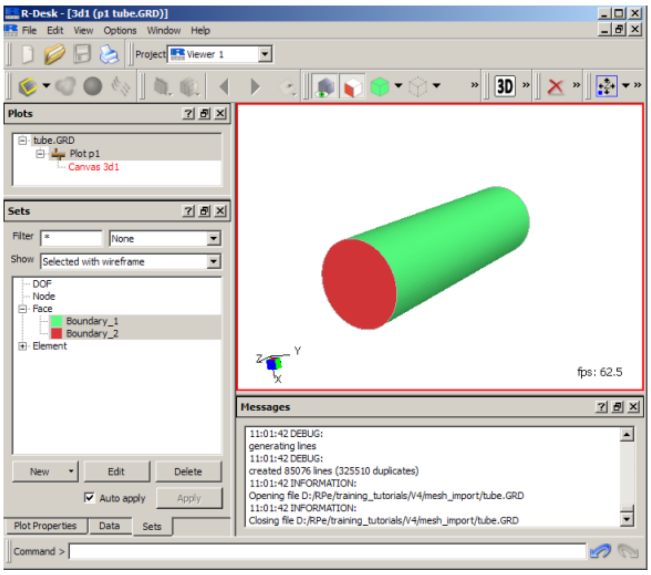
19.4.2 Simple Example

In this example case a simple tetrahedral mesh of a tube is imported from an Ideas format .unv file. The file tube.unv can be found following the link above.

Firstly convert the external mesh file to the native .GRD file using vpre. This can be done either on the command line or from the R-Desk

The .GRD file can then be read in to R-Desk so the the the boundary regions can be defined. For GRD file face sets are used to define the boundary regions.

In the example case the .unv file included some node sets. These are converted to boundary definitions by vpre and applied to the GRD file. For imported files that do not contain supported boundary definitions, all faces will be assign the Boundary 1. The imported sets are shown in Figure 19.91

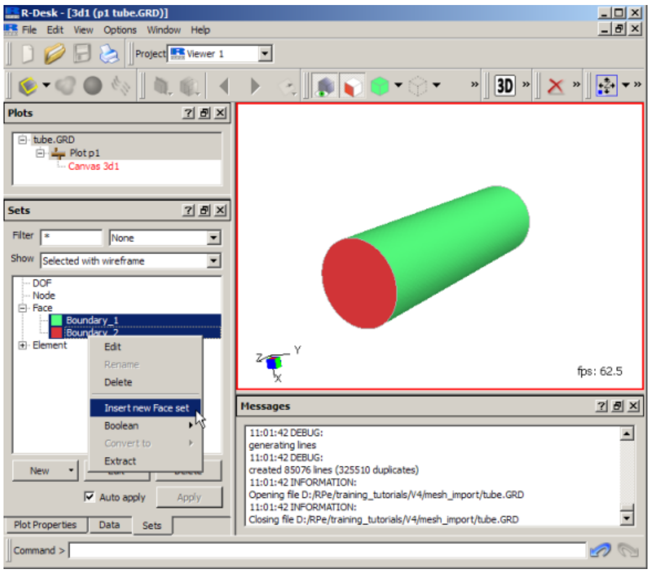


```
11:01:42 DEBUG:
generating lines
11:01:42 DEBUG:
created 85076 lines (325510 duplicates)
11:01:42 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:01:42 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.91: The GRD file when loaded into R-Desk

19.4.2.1 Creating Boundary Regions

In this case we wish to define each ends of the tube to be separate boundaries with all the remaining faces to be in an additional boundary. As such a third boundary face set needs to be created. This can be done by 'right-clicking' on the 'face' entry in the set tree and selecting 'Create new face set'. Figure 19.92.

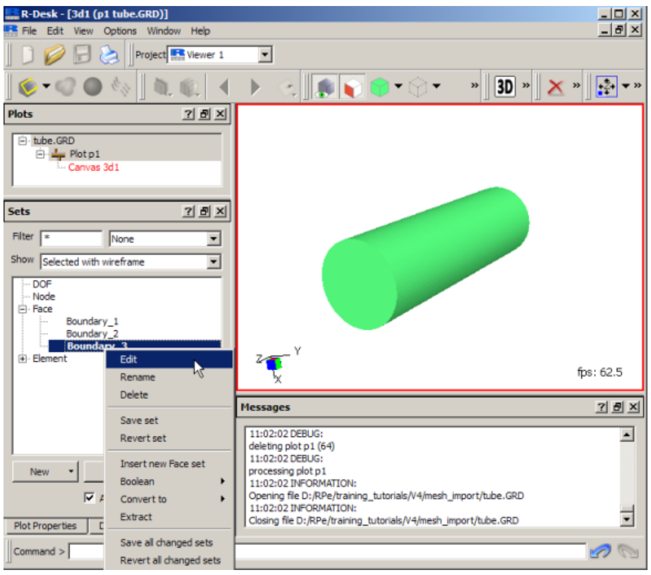


```
11:01:42 DEBUG:
generating lines
11:01:42 DEBUG:
created 85076 lines (325510 duplicates)
11:01:42 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:01:42 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.92: Creating a new face set

19.4.2.2 Editing Sets

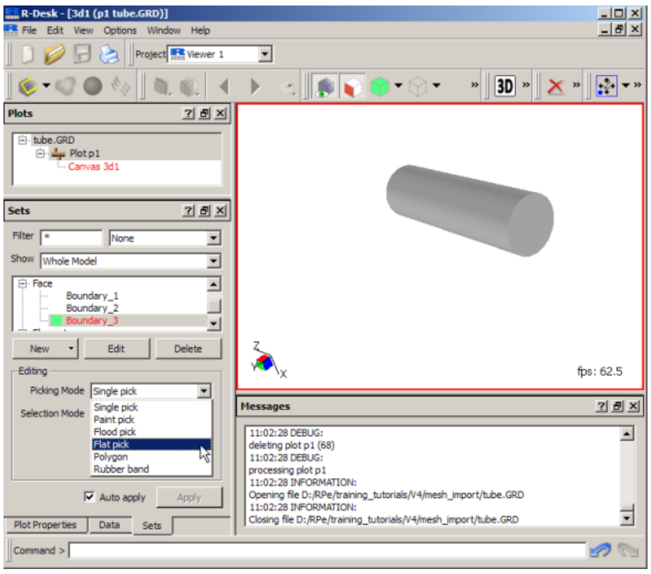
Once the face set has been created then the boundary faces that will make up the region should be painted. This is done by editing the face set. Either right-click the set name and select 'edit' or select the set and then press the edit button. The set being edited will be written in red and the edit options will be displayed in the sets panel Figure 19.93. Different options are available for editing sets, faces can be selected, deselected or toggled.



```
11:02:02 DEBUG:
deleting plot p1 (64)
11:02:02 DEBUG:
processing plot p1
11:02:02 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:02:02 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.93: Editing the face set

Different picking options are available for editing sets, Figure 19.94



```
11:02:28 DEBUG:
deleting plot p1 (68)
11:02:28 DEBUG:
processing plot p1
11:02:28 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:02:28 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.94: Pick options available for set editing

Single Pick In this mode, faces are picked one at a time. To pick a face, press Shift+Left-Click, and when the mouse is released the face under the cursor will be picked (shown by colouring it red) and added to the set.

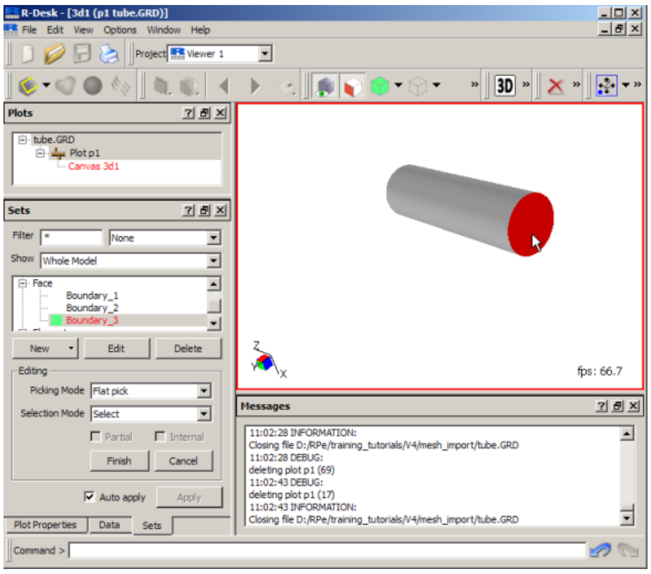
Paint Pick In this mode, faces are picked as if they are being painted over. To pick a face, press Shift+Left-Click, and then drag the cursor across the model. As soon as the cursor moves over an unselected face it will be selected and added to the current set.

Flood Pick In this mode, faces are picked within the bounds of an existing selection, e.g. if we draw a circle around the circumference of our port, we can then flood pick the inner faces. To flood pick, press Shift+Left-Click and then drag the cursor across the model; as soon as the cursor moves over an unselected face it will be added to the current set.

Flat Pick In this mode, all faces on the same surface are picked up to an edge, where the edge angle is defined in the Preferences dialog. To flat pick, press Shift+Left-Click, and when the mouse is released all faces on the same surface under the cursor will be added to the current set

Polygon This mode involves tracing out a polygon, and all faces within the polygon become selected. To start the polygon, Shift+Left-Click anywhere in the 3D canvas (if the canvas is not already selected then it may take two clicks to begin the polygon). The first point of the polygon is shown as a square, subsequent Shift+Left-Clicks add extra nodes to the polygon. To finish the polygon requires a Shift+Double Left-Click

RubberBand Rubber band picking is similar in concept to Polygon picking, except that a rectangular box is traced out with the mouse instead of a polygon



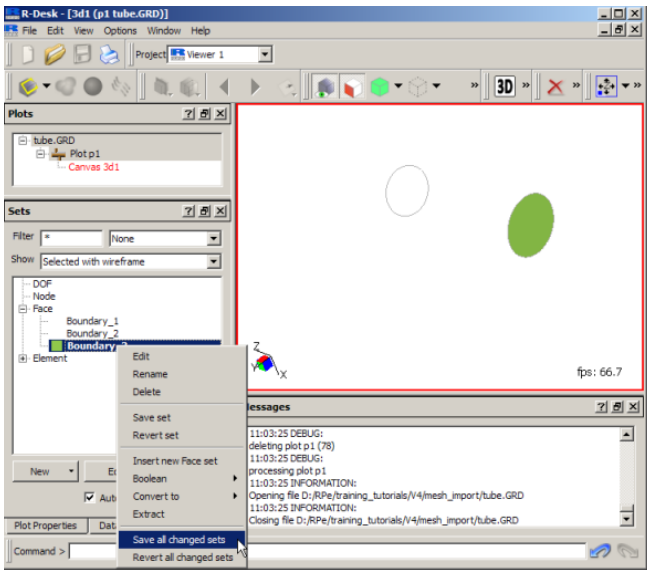
Filter: * None
Show: Whole Model
Picking Mode: Flat pick
Selection Mode: Select

```
11:02:28 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:02:28 DEBUG:
deleting plot p1 (69)
11:02:43 DEBUG:
deleting plot p1 (17)
11:02:43 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.95: Use Flat Picking to select the required face

In this case Flat Pick will be used to select the other end of the tube to 'boundary 2'. Rotate the model to see the other end of the model. Shift and left-click on a face in the required region to paint the entire end of the tube. Figure 19.95.

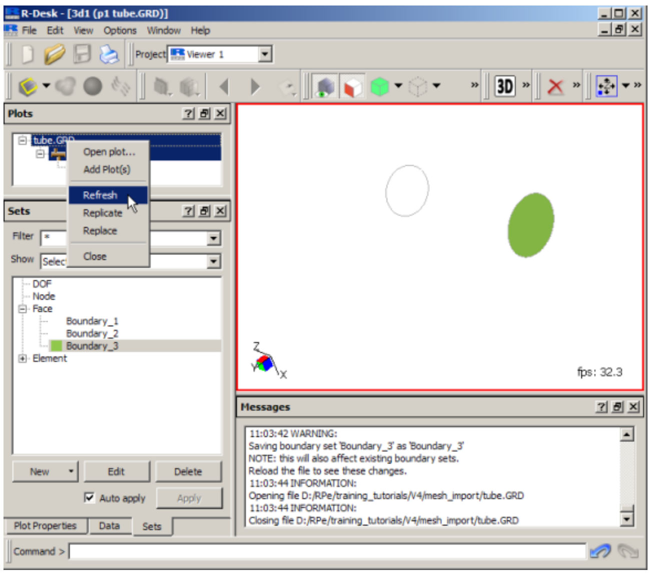
Once the new face sets have been defined then these new regions need to be saved to the GRD file. Right click on one of the face sets and select 'save all sets', Figure 19.96.



```
11:03:25 DEBUG:
deleting plot p1 (78)
11:03:25 DEBUG:
processing plot p1
11:03:25 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:03:25 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.96: Saving the sets to the GRD file

One thing to note is that face sets are non-exclusive, in that a face be defined to belong to more than one face set. However this is not the case for the boundary regions on a mesh. If a face is found to belong to a number of different 'boundary' sets the highest numbered boundary region will take precedence. To avoid confusion it is recommended that this situation is avoided. Use the deselect function of the set editing tools to remove unwanted faces from boundary regions. Additionally all faces should belong to a boundary region and should not be left unpainted.



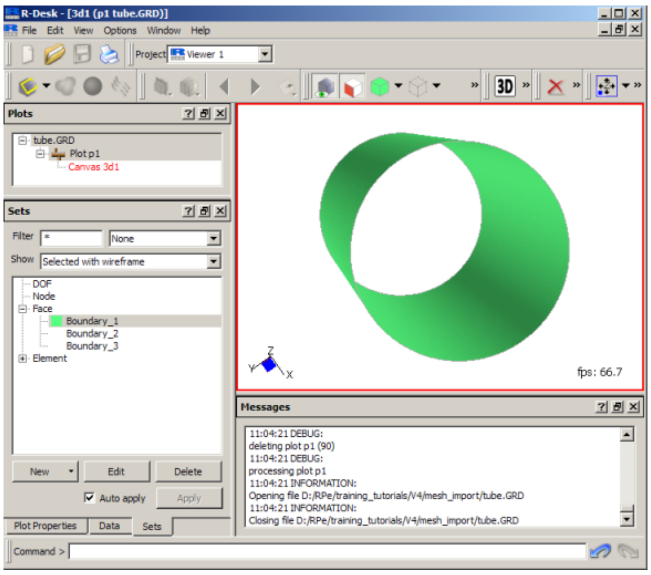
```
11:03:42 WARNING:
Saving boundary set 'Boundary_3' as 'Boundary_3'
NOTE: this will also affect existing boundary sets.
Reload the file to see these changes.
11:03:44 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:03:44 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.97: Reloading the GRD file

Refresh the saved GRD file to visualise the the final boundary region definitions. Right-Click on the file name in the plot tree and select 'Refresh', Figure 19.97.

Select the different face sets to check that they are defined correctly. Figure 19.98.

Once the boundaries have been defined correctly, run vpre on the GRD file. This will reorder the



```
11:04:21 DEBUG:
deleting plot p1 (90)
11:04:21 DEBUG:
processing plot p1
11:04:21 INFORMATION:
Opening file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
11:04:21 INFORMATION:
Closing file D:/RPe/training_tutorials/V4/mesh_import/tube.GRD
```

Figure 19.98: Viewing the defined sets on the final GRD file

faces according to the new boundary region definitions. Once this has been done then the mesh can be imported as normal to set-up the .inp file in a VECTIS project.

Other Notes: In the case of multi-domain simulations. Each domain should be converted separately. The boundaries and interface regions for each domain then needs to be identified before joining all the domains using *vpre* in the standard way.

19.4.2.3 Importing grid file data into solver setup

Now that the boundaries have been defined the computational grid can be imported into the solver setup. In a VECTIS session in *R-Desk* click on 'solver setup' in the solver setup tree . Browse to the .GRD generated previously. Click on extract and then OK . The boundaries defined in the file will then be read and added to the solver setup tree. Check the Show mesh preview toggle and the imported grid file will be displayed.

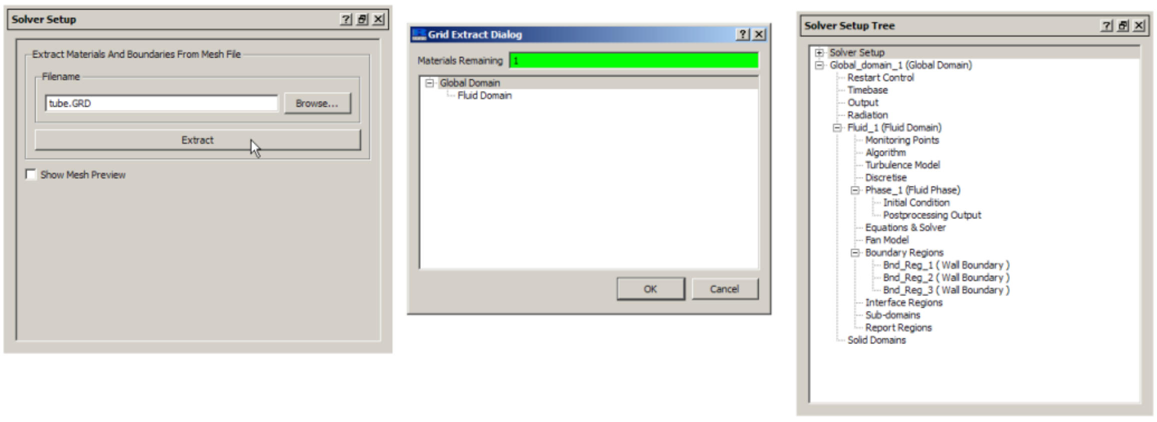


Figure 19.99: Importing the generated computational mesh into the solver setup

Subsequent selection of the boundaries regions in the solver setup tree will then highlight the corresponding regions on the model in the mesh preview.

Bibliography

Absi, R. and Bennacer, R. [2006], A New Wall Function for the Turbulent Kinetic Energy *k*, in Y.N. K. Hanjalic and S. Jakirlic, eds., *Turbulence, Heat and Mass Transfer*, 5, Begell House, Inc. 14.4

Baldwin, B. and Lomax, H. [1978], Thin-Layer Approximation and Algebraic Model for Separated Turbulent Flows, *AIAA Paper 78-257*. 9.2.1

Barre, S., Bonnet, J.P., Gatski, T. and Sandham, N. [2002], Compressible, High Speed Flows, in B. Launder and N. Sandham, eds., *Closure Strategies for Turbulent and Transitional Flows*, Cambridge University Press, Cambridge. 6.4, 9.3

Barth, T.J. [1994], Aspects of Unstructured Grids and Finite–Volume Solvers for the Euler and Navier–Stokes Equations, in *Computational Fluid Dynamics*, lecture Series von Karman Institute for Fluid Dynamics. 14.1.4

Benzi, M. [2002], Preconditioning Techniques for Large Linear Systems: A Survey, *Journal of Computational Physics*, vol. 182 pp. 418–477. 14.2.6

Bo, T. [2004], CFD Homogeneous Mixing Flow Modelling to Simulate Subcooled Nucleate Boiling Flow, in *SAE International*, 2004–01–1512. 12.4.1.1

Brackbill, J., Kothe, D. and Zemach, C. [1992], A Continuum Method for Modelling Surface Tension, *Journal of Computational Physics*, vol. 100 pp. 335–354. 12.3.2

Bradshaw, P. [1994], Turbulence: The Chief Outstanding Difficulty of Our Subject, *Experiments in Fluids*, vol. 16. 9.1, 9.3

Davydov, B.I. [1961], On Statistical Dynamics of an Incompressible Turbulent Fluid, *Soviet Physics - Doklady, Fluid Mechanics*, vol. 6 pp. 10–12. 9.2.1

de Lemos, M.J.S. [2005], Fundamentals of the Double–Decomposition Concept for Turbulent Transport in Permeable Media, *Mat.–Wiss. u. Werkstofftech.*, vol. 36, no. 10 pp. 586–593. 11.1.2

de Lemos, M.J.S. and Pedras, M.H.J. [2001], Recent Mathematical Models for Turbulent Flow in Saturated Rigid Porous Media, *ASME Journal of Fluids Engineering*, vol. 123 pp. 935–940. 11.1.3, 11.1.3

Demirdzic, I., Lilek, Z. and Peric, M. [1993], A Collocated Finite Volume Method for Predicting Flows At All Speeds, *Int. Journal for Numerical Methods in Fluids*, vol. 16 pp. 1029–1050. 14, 14.2.3, 14.2.4

Demirdzic, I. and Muzaferija, S. [1995], Numerical Method for Coupled Fluid Flow, Heat Transfer and Stress Analysis Using Unstructured Moving Meshes with Cells of Arbitrary Topology, *Comp. Methods Appl. Mech. Eng.*, vol. 125 pp. 235–255. 14.1.3

Drew, D. [1992], Analytical Modelling of Two-Phase Flow, Boiling Heat Transfer, Elsevier Science Publ. 12.2

Durbin, P. [1991], Near–Wall Turbulence Closure Modeling without 'Dumping Functions', *Theoretical and Computational Fluid Dynamics*, vol. 3, no. 1 pp. 1–13. 9.2.1, 9.3, 9.4.1, 9.5

Durbin, P.A. [1996], On the $k-\varepsilon$ Stagnation Point Anomaly, *Int. J. Heat and Fluid Flow*, vol. 17, no. 1 pp. 89–90. 9.3.3, 9.5, 9.5

Durbin, P.A. [2009], Limiters and Wall Treatments in Applied Turbulence Modelling, *Fluid Dynamics Research*, vol. 41 pp. 1–18. 9.5

Esch, T. and Menter, F.R. [2003], Heat Transfer Predictions Based on Two–Equation Turbulence Models with Advanced Wall Treatment, in Y. K. Hanjalic and M. Tummers, eds., *Turbulence, Heat and Mass Transfer*, 4, Begell House, Inc., pp. 663–640. 9.4.3

Ferziger, J. and Peric, M. [1997], *Computational Methods for Fluid Dynamics*, Springer, Berlin. 14.1.3, 14.1.5.1, 14.2.2, 14.2.3

Gaskell, P.H. and Lau, A.K. [1988], Curvature Compensated Convective Transport: SMART, A New Boundedness Preserving Transport Algorithm, *International Journal for Numerical Methods in Fluids*, vol. 8 pp. 617–641. 14.1.3.1, 14.1.3.1, 14.1.3.1

Gatski, T.B. and Rumsey, C.L. [2002], Linear and Nonlinear Eddy Viscosity Models, in B. Launder and N. Sandham, eds., *Closure Strategies for Turbulent and Transitional Flows*, Cambridge University Press, Cambridge. 9.2.1

George, A. and Liu, J. [1981], *Computer Solution of Large Sparse Positive Definite Matrices*, Prentice Hal. 14.2.6

George, W.K. [2007], Is There a Universal Log Law for Turbulent Wall– Bounded Flows, *Phil. Trans. Royal Soc. A*, vol. 365 pp. 789–806. *

Hanjalic, K. [1970], *Two-Dimensional Flow in an Axisymmetric Channel*, ph.D. Thesis, University of London. 9.2.1

Hanjalic, K. [1994], Advanced Turbulence Closure Models: A View on the Current Status and Future Prospects, *Int. J. Heat & Fluid Flow*, vol. 15 pp. 178–203. 9.3.1, 9.6

Hanjalic, K. [2005], Will RANS Survive LES? A View of Perspectives, *Journal of Fluids Engineering*, vol. 127 pp. 831–839. 9.1

Hanjalic, K. and Launder, B.E. [1972], A Reynolds Stress Model of Turbulence and Its Application to Thin Shear Flows, *Journal of Fluid Mechanics*, vol. 52 pp. 609–638. 9.2.1

Harlow, F. and Nakayama [1967], Turbulence Transport Equations, *Physics of Fluids*, vol. 10 pp. 2323–2332. 9.2.1

Hinze, J.O. [1975], *Turbulence*, 2nd ed., McGraw–Hill, New York. 6.3.1

Hirt, C. and Nicholls, B. [1981], Volume of Fluid (VOF) Method for Dynamics of Free Boundaries, *J. Comput. Phys.*, vol. 39 pp. 201–221. 12.3

Huang, P., Coleman, G. and Bradshaw, P. [1995], Compressible Turbulent Channel Flows: DNS Results and Modelling, *Journal of Fluid Mechanics*, vol. 305 pp. 185–218. 6.4, 6.4

Ishii, M. [1975], *Thermo-Fluid Dynamic Theory of Two-Phase Flow*, Eyrolles, Paris. 12.1, 12.2

Jayatillaka, C.V. [1969], The Influence of Prandtl Number and Surface Roughness on the Resistance of the Laminar Sublayer to Momentum and Heat Transfer, *Prog. Heat Mass Transfer*, vol. 1, p. 193. 9.4.2

Jones, W. and Launder, B. [1972], The Prediction of Laminarization with a Two- Equation Model of Turbulence, *International Journal of Heat and Mass Transfer*, vol. 15 pp. 301–314. 9.2.1

Kader, B.A. [1981], Temperature and Concentration profiles in Fully Turbulent Boundary Layers, *Int. J. Heat Mass Transfer*, vol. 24, no. 9 pp. 1541–1544. 9.4.2

Kalitzin, G. and Iaccarino, G. [2003], Toward Immersed Boundary Simulation of High Reynolds Number Flows, *Center for Turbulence Research Annual Research Briefs* pp. 369–378, Stanford University. 14.4

Kalitzin, G., Medic, G., Iaccarino, G. and Durbin, P. [2005], Near–Wall Behavior of RANS Turbulence Models and Implications for Wall Functions, *Journal of Computational Physics*, vol. 204 pp. 265–291. 14.4

Karki, K.C. and Patankar, S.V. [1989], Pressure Based Calculation Procedure for Viscous Flows At All Speeds in Arbitrary Configurations, *AIAA Paper 86–0207*, vol. 27 pp. 1167–1174. 14

Kenning, D.B.R. and Victor, H.T. [1981], Fully-developed nucleate boiling: overlap of areas of influence and interference between bubble sites, *Int. J. Heat Mass Transfer*, vol. 24 pp. 1025–1032. 12.4.1.2

Khosla, P.K. and Rubin, S.G. [1974], A Diagonally Dominant Second–Order Accurate Implicit Scheme, *Computers & Fluids*, vol. 2 pp. 207–209. 14.1.5.2

Kim, S.E. and Choudhury, D. [1995], A Near–Wall Treatment Using Wall Function Sensitized to Pressure Gradient, in *ASME Symp. Separated and Complex Flows*, FED–Vol. 217, pp. 273–280. 9.4.2

Kunz, R., Siebert, B., Cope, W., Foster, N., Antal, S. and Ettorre, S. [1998], A Coupled Phasic Exchange Algorithm for Three-Dimensional Multi-Field Analysis of Heated Flows with Mass Transfer, *Computers & Fluids*, vol. 27 p. 741. 12.2

Kurul, N. and Podowski, M. [1990], Multidimensional effects in forced convection subcooled boiling, in *Proceedings of the 9th International Heat Transfer Conference, Jerusalem, Israel*, 1-BO-04, pp. 21–26. 12.4.1.2

Lahey, R. and Drew, D. [2001], The Analysis of Two-Phase Flow and Heat Transfer Using a Multidimensional, Four Field, Two-Fluid Model, *Nuclear Engineering and Design*, vol. 204 pp. 29–44. (document), 12.1

Lamb, H. [1932], *Hydrodynamics*, 6th ed., University Press, Cambridge. 9.1

Landahl, M. and Mollo-Christensen, E. [1986], *Turbulence and Random Processes in Fluid Mechanics*, Cambridge University Press, New York. 6.3.1

Launder, B.E. and Spalding, D.B. [1974], The Numerical Computation of Turbulent Flows, *Computer Methods in Applied Mechanics and Engineering*, vol. 3 pp. 269–289. 9.2.1, 9.4.2

Leonard, B.P. [1988], Simple High–Accuracy Resolution Program for Convective Modelling of Discontinuities, *International Journal for Numerical Methods in Fluids*, vol. 8 pp. 1291–1318. 14.1.3.1

Leschziner, M.A., Batten, P. and Loyau, H. [2000], Modelling Shock–Affected Near–Wall Flows with Anisotropy–Resolving Turbulence Closures, *International Journal of Heat and Fluid Flow*, vol. 21 pp. 239–251. 9.3

Meijerink, J.A. and Van Der Vorst, H.A. [1981], Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations As They Occur in Practical Problems, *Journal of Computational Physics*, vol. 44 pp. 134–155. 14.2.6

Menter, F. [1994], Two Equation Eddy–Viscosity Turbulence Models for Engineering Applications, *AIAA Journal*, vol. 32 pp. 1598–1605. 9.2.1

Moran, M.J. and Shapiro, H.N. [1992], *Fundamentals of Engineering Thermodynamics*, 2nd ed., John Wiley & Sons, Inc., New York. 6.2.1, 8.3.2

Moser, R., Kim, J. and Mansour, N. [1999], Direct numerical simulation of turbulent channel flow up to Re_τ = 590, *Physics of Fluids*, vol. 11, no. 4 pp. –943. (document), 9.4.4, 9.2, 9.3

Murthy, J.Y. and Mathur, S.R. [1998], A Conservative Numerical Scheme for the Energy Equation, *ASME Journal of Heat Transfer*, vol. 120 pp. 1081–1086. 14.1.6, 14.1.6

Muzaferija, S. [1994], Adaptive Finite Volume Method for Flow Predictions Using Unstructured Meshes and Multigrid Approach, PhD dissertation, University of London. 14.1.4

Nakayama, A. [2008], Theory of Porous Media and Its Numerical Applications to Engineering Problems, in *Proceedings CHT-08: International Symp. Advances in Computational Heat Transfer*, Marrakech, Morocco. 11.1.3, 11.1.3

Nakayama, A. and Kuwahara, F. [1999], A Macroscopic Turbulence Model for Flow in a Porous Medium, *ASME Journal of Fluids Engineering*, vol. 121 pp. 427–433. 11.1.3

Patankar, S.V. [1980], *Numerical Heat Transfer and Fluid Flow*, McGraw–Hill, New York. 10.2.1, 14, 14.1.5.4, 14.1.6, 14.2.6

Pedras, M.H.J. and de Lemos, M.J.S. [2001], Macroscopic Turbulence Modeling for Incompressible Flow Through Undeformable Porous Media, *Int. J. Heat Mass Transfer*, vol. 44 pp. 1081–1093. 11.1.3

Popovac, M. and Hanjalic, K. [2007], Compound Wall Treatment for RANS Computations for Complex Turbulent Flows and Heat Transfer, *Flow, Turbulence and Combustion*, vol. 78, no. 2 pp. 177–202. 9.4.4, 9.6

Przulj, V. [2009], Pragmatic Wall Treatment for RANS Simulations on Cartesian Cut–Cell Grids, in K. Hanjalic and Y. Nagano, eds., *6th Int. Symp. Turbulence, Heat and Mass Transfer*, Rome, Italy. 9.4.4

Przulj, V. and Basara, B. [2001], Bounded Convection Schemes for Unstructured Grids, *AIAA paper 2001–2593*. 14.1.3

Przulj, V. and Basara, B. [2002], A SIMPLE –Based Control Volume Method for Compressible Flows on Arbitrary Grids, *AIAA 2002 –3289*. 14, 14.1.4

Ranz, W.E. and Marshall, W.R. [1952], Evaporation from drops, *Heat Transfer*, vol. 48 pp. 142–180. 12.4.1.2

Rayleigh, L. [1917], On the pressure developed in a liquid during the collapse of a spherical cavity, *Phil. Mag.*, vol. 34 pp. 94–98. 12.4.2

Rhie, C.M. and Chow, W.L. [1983], Numerical Study of the Turbulent Flow Past an Airfoil with Trailing Edge Separation, *AIAA Journal*, vol. 21 pp. 1525–1532. 14.2.2

Rodi, W., Mansour, N.M. and Michelassi, V. [1993], One–Equation Near–wall Turbulence Modeling With the Aid of Direct Simulation Data, *ASME Journal of Fluids Engineering*, vol. 115 pp. 196–205. 9.4.4

Rohsenow, W.M. [1952], A Method of Correlation Heat Transfer Data for Surface Boiling of Liquid, *Trans. ASME*, vol. 72 pp. 1025–1032. 12.4.1.1

Rung, T., Lubcke, H.M. and Thiele, F. [2000], Universal Wall–Boundary Conditions for Turbulence Transport Models, *Zetschrift fur Angewandte Mathematik und Mechanik* pp. 1756–1758. 9.4.2, 9.4.4

Saffman, P.G. [1970], A Model for Inhomogeneous Turbulent Flow, *Proc. Roy. Soc. London A*, vol. 317 pp. 417–433. 9.2.1

Schnerr, G. and Suaer, J. [2001], Physical and numerical modeling of unsteady cavitation dynamics, in *ICMF2001*, New Orleans, USA. 12.4.2.3

Shih, T.H., Liou, W.W., Shabbir, A., Yang, Z. and Zhu, J. [1995], A New k- Eddy- Viscosity Model for High Reynolds Number Turbulent Flows - Model Development and Validation, *Computers Fluids*, vol. 24, no. 3 pp. 227–238. 9.2.1

Singhal, A., Athavale, M., Li, H. and Jiang, Y. [2002], Mathematical basis and validation of the full cavitation model, *Journal of Fluids Engineering*, vol. 124 pp. 617–624. 12.4.2.1

Slattery, J.C. [1999], *Advanced Transport Phenomena*, Cambridge University Press, Cambridge. 11.1.1

Sleijpen, G. and Fokkema, D.R. [1993], BiCGstab(l) for Linear Equations Involving Unsummetric Matrices with Complex Spectrum, *Electronic Trans. Numer. Anal.*, vol. 1 pp. 11–32. 14.2.6

Smagorinsky, J. [1963], General Circulation Experiments with the Primitive Equations I. The Basic Experiment, *Mon. Weather Rev.*, vol. 91 pp. 99–164. 9.2.1

Spalart, P. [1988], Direct simulation of a turbulent boundary layer up to $Re_\theta = 1410$, *Journal of Fluid Mechanics*, vol. 187 pp. 61–98. (document), 9.4.4, 9.2

Spalart, P.R. and Allmaras, S.R. [1994], A One–Equation Model for Aerodynamic Flows, *La Recherche Aerospatiale*, vol. 1 pp. 5–21. 9.2.1

Spalding, D.B. [1991], Kolmogorov's Two-Equation Model of Turbulence, *Proc. Roy. Soc. London A*, vol. 434 pp. 211–216. 9.2.1

Speziale, C.G. [1991], Analytical Methods for the Development of Reynolds-Stress Closures in Turbulence, *Annual Review of Fluid Mechanics*, vol. 23 pp. 107–157. 9.2.1

Speziale, C.G. [1996], Modeling of Turbulent Transport Equations, in T. Gatski, M. Hussaini and J. Lumley, eds., *Simulation and Modeling of Turbulent Flows*, Oxford University Press, Oxford. 6.4, 6.4, 9.3, 9.3.1

Sweby, P.K. [1984], High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws, *SIAM Journal Numer. Anal.*, vol. 21 pp. 995–1011. 14.1.3.1

Tennekes, H. and Lumley, J. [1986], *A First Course in Turbulence*, MIt Press, Cambridge, MA. 6.3.1

Tolubinsky, V. and Konstanchuk, D. [1970], Vapor bubbles growth rate and heat transfer intensity at subcooled water boiling, *Heat Transfer*, vol. 5. 12.4.1.2

Ubbink, O. [1997], *Numerical Prediction of Two-Fluid Systems with Sharp Interfaces*, ph.D. Thesis, Imperial College, University of London,. 12.3.2

Van Der Vorst, H.A. [1992], Bi–CGSTAB: A Fast and Smoothly Converging Variant of Bi–CG for the Solution of Nonsymmetric Linear Systems, *SIAM J. Scientific Computing*, vol. 13 pp. 631–644. 14.2.6

Vandromme, D. [1993], Turbulence Modeling for Compressible Flows and Implementation in Navier–Stokes Solvers, in *Introduction to the Modeling of Turbulence*, Lecture Series, von Karman Institute for Fluid Dynamics. 6.4

Veynante, V. and Vervisch, L. [2002], Turbulent combustion modeling, *Progress in Energy and Combustion Science*, vol. 28 pp. 193–266. 6.4

Wark, K. [1983], *Thermodynamics*, 4th ed., McGraw–Hill, New York. 8.3.2, ??

Watterson, J.K., Dawes, W.N., Savill, A.M. and J., W.A. [1999], Predicting Turbulent Flow in a Staggered Tube Bundle, *Int. J. Heat & Fluid Flow*, vol. 20 pp. 581–591. 9.5

Whitaker, S. [1999], *The Method of Volume Averaging*, Kluwer Academic Publishers, Dordrecht. 11.1.1, 11.1.1

Wilcox, D. [1998], *Turbulence Modeling for CFD*, 2nd ed., DCW Industries, Inc., La Canada, California. 6.4, 9.2.1

Wolfshtein, M. [1969], The Velocity and Temperature Distribution in One–Dimensional Flow with Turbulence Augmentation and Pressure Gradient, *Int. J. Heat Mass Transfer*, vol. 12 pp. 301–318. 9.4.2, 9.4.4

Yakhot, V., Orszag, S.A., Thangam, S., Gatski, T.B. and Speziale, C.G. [1992], Development of Turbulence Models for Shear Flows by a Double Expansion Technique, *Physics of Fluids A*, vol. 4 pp. 1510–1520. 9.2.1, 9.3.2

Yakhot, V. and Orszag, S. [1986], Renormalization Group Analysis of Turbulence. I. Basic Theory, *Journal of Scientific Computing*, vol. 1 pp. 3–50. 9.2.1, 9.3.2

Yang, Z. and Shih, T. [1993], New Time Scale Based $k-\varepsilon$ Model for Near–Wall Turbulence, *AIAA Journal*, vol. 31, no. 7 pp. 1191–1198. 9.3, 9.5

Yao, Y.F., Savill, A.M., Sandham, N.D. and Dawes, W.N. [2002], Simulation and Modelling of Turbulent Trailing–Edge Flow, *Flow, Turbulence and Combustion*, vol. 68 pp. 313–333. 9.5

Zwart, P., Gerber, A. and Belamri, T. [2004], A two-phase flow model for predicting cavitation dynamics, in *ICMF2004*, Yokohoma, Japan. 12.4.2.2